IDENTIFICATION

DIAGNOSTIC CODE:	DFCIC
PRODUCT NAME:	DFCICCO CI Cluster Test Protocol Responder
VERSION:	3
DATE RELEASED:	August 27, 1985
MAINTAINED BY:	Large Systems Diagnostics
AUTHOR:	Ron Stravinski
UPDATE AUTHOR:	Gregory A. Scott

COPYRIGHT (C) 1984, 1985

DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASS.

THIS SOFTWARE IS FURNISHED UNDER A LICENSE FOR USE ONLY ON A SINGLE COMPUTER SYSTEM AND MAY BE COPIED ONLY WITH THE INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE, OR ANY OTHER COPIES THEREOF, MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY OTHER PERSON EXCEPT FOR USE ON SUCH SYSTEM AND TO ONE WHO AGREES TO THESE LICENSE TERMS. TITLE TO AND OWNERSHIP OF THE SOFTWARE SHALL AT ALL TIMES REMAIN IN DIGITAL EQUIPMENT CORPORATION.

THE INFORMATION IN THIS DOCUMENT IS SUBJECT TO CHANGE WITHOUT NOTICE AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT CORPORATION.

DIGITAL EQUIPMENT CORPORATION ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS SOFTWARE IN EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL EQUIPMENT CORPORATION.

Table of Contents

		Page
1.0	Overview of Diagnostic Product	1
1.1	Abstract	1
1.2	Users and Uses	1
1.3	Pre-requisite software	1
2.0	Product Goals/Non-Goals	2
2.1	Performance goals	2
2.1.1	Program Size	2
2.1.2	Program Run Time	2
2.1.3	Exec Mode Capability	2
2.1.4	User Mode Capability	2
2.1.5	Fault Detection	2
2.1.6	Fault Isolation	3
2.1.7	General User Interface Philosophy	3
2.2	Compatibility Goals	3
2.2.1	CPU Family	3
2.2.2	Standard Diagnostic Support Software	3
2.2.3	Operating systems (TOPS-10)	3
2.2.4	Operating systems (TOPS-20)	4
2.2.5	Spear	4
2.3	Failsoft Goals	4
2.3.1	CPU/Memory Failures	4
2.3.2	Power fail/restart	4
2.4	Restrictions	4
2.4.1	User Mode Privileges	4
2.4.2	User Mode Priority Interrupts	4
2.5	Non-Goals	5
3.0	Requirements	5
3.1	Hardware Design Requirements	5
3.2	Run Time Requirements	5
3.2.1	CPU family	5
3.2.2	Memory	5
3.2.3	Load Device	5
3.2.4	User Interface	6
3.2.5	Operating Systems	6
4.0	Functional Description (Overview)	6
4.1	Error Reporting Philosophy	6
4.1.1	Error Reporting of Hardware Detected Errors	7

4.1.2 Reporting to SPEAR 7

4.2 Response Decoding 7

4.3 General Test Philosophy 7
4.3.1 Responder Data Structure 7
4.3.2 Delay/Timeout Data Structure 10
4.3.3 Third Party Data Structure 10
4.3.4 Request execution scheme 12

4.4 Command/Response Descriptions 13
4.4.1 Responder Function Set Request 13
4.4.2 Buffer Map Request 13
4.4.3 Buffer Unmap Request 14
4.4.4 Move Buffer Request 14
4.4.5 Generate Message Request 14
4.4.6 Generate Datagram Request 15
4.4.7 Generate Reset Request 15
4.4.8 Generate Start Request 15
4.4.9 Start/Stop Unsolicited Activity 15
4.4.10 Configuration Data Request 16
4.4.11 Counter Read Request 16
4.4.12 Connect Request 16
4.4.13 Listener Disconnect Request 16

5.0 Program Interfaces 16

5.1 Diagnostic Software Interfaces 17

5.2 Operating System Interfaces (TOPS-10) 17

5.3 Operating System Interfaces (TOPS-20) 17

6.0 Appendices 18

6.1 Appendix A (Figures) 18

6.2 Appendix B (CTP Messages) 19
6.2.1 CTP Message Formats 19
6.2.2 Responder Function Set Messages 19
6.2.3 Buffer Map Messages 21
6.2.4 Buffer Unmap Messages 24
6.2.5 Move Buffer Messages 26
6.2.6 Generate Message Messages 29
6.2.7 Generate Datagram Messages 31
6.2.8 Generate Reset Messages 33
6.2.9 Generate Start Messages 35
6.2.10 Start/Stop Unsolicited Activity Messages . . . 37
6.2.11 Configuration Data Messages 39
6.2.12 Counter Read Messages 41
6.2.13 Connect Messages 44
6.2.14 Listener Disconnect Messages 45

1.0 Overview of Diagnostic Product

1.1 Abstract

The CI Cluster Test Protocol Responder is the slave portion of a master/slave pair of cooperating programs. Its function is to receive and act according to instructions from the master process referred to as a Controller Process. In the CI environment, the Controller Process may be the CI Node Tester (CINT) or a copy of the CI Exerciser Program (CIE) running in some CI Node.

The CI Responder is a user mode only program that will run under the TOPS-20 Operating System (release 6.0 or newer) or the TOPS-10 Operating System (release 7.03 or newer). The Responder will run during normal timesharing as an OPERATOR or [1,2] job and does not require exclusive use of any system resources.

Command and Response Packets are passed between Controller and Responder Processes using the Cluster Test Protocol (CTP). The System Communication Service (SCS) of the Operating System will be used to send CTP messages over the CI.

1.2 Users and Uses

The Responder will be used by Field Service and Manufacturing in the Dock Merge verification process along with the CIE. Manufacturing may do Quality Control checks at certain intervals in the manufacturing process in which they would use the Responder with the CINT. Since the Responder will be run as an OPERATOR job, it will be accessible to anyone in a remote node who would like to checkout the CI, for example a Field Service person. Engineering will use the Responder in development of current and future CI Products.

1.3 Pre-requisite software

- CPU Diagnostics
- Memory Diagnostics
- I/O Box Diagnostics
- CI Port Hardware Diagnostics
- TOPS-20 Operating System or
- TOPS-10 Operating System

2.0 Product Goals/Non-Goals

2.1 Performance goals

To be able to execute, with a high degree of predictability, the commands received, and to respond with usable data for the CI Exerciser or CINT. The Responder is designed to return accurate status information to the controlling node.

2.1.1 Program Size

It is expected that the Responder will execute in less than 30 pages. The Responder has the ability to have 32 virtual connections open at any one time.

2.1.2 Program Run Time

The Responder is always be running in "Listen Mode", awaiting a connection from some controlling process. Run time in the traditional sense does not apply. Once a connection is established, the amount of the work done by the Responder is determined totally by the Controlling Process.

2.1.3 Exec Mode Capability

The responder does not run in Exec Mode.

2.1.4 User Mode Capability

The responder runs in a normal timesharing job.

2.1.5 Fault Detection

The Responder has the ability to detect 2 types of errors.

1. Failure to do what the Controller Requested.
2. A JSYS or UUO failure when trying to send a response or perform some other monitor assisted operation.

2.1.6 Fault Isolation

The Responder has no built in isolation capability. Any functional fault detected will be passed back to the CIE for analysis and reporting. The Exerciser will then isolate to the node and identify the function that was being performed when the error occurred. Information about broken connections or error returns from a JSYS or UUO (monitor call) within the Responder will not be sent to the controller.

2.1.7 General User Interface Philosophy

There is not be any user interface to this Program. All commands and responses will be through an SCA Connection with a Controller.

2.2 Compatibility Goals

The Responder will be able to respond to CTP Commands received from any CI node (HSC50, Vax, KL10, CINT).

2.2.1 CPU Family

The responder is a user program and is monitor dependent rather than hardware dependent. It will run on any 36 bit CPU provided the TOPS-10/TOPS-20 monitor supports the CI.

2.2.2 Standard Diagnostic Support Software

The Responder does not use any of the Standard Diagnostic Support Software. It is a self contained program with an ".EXE" extension. DIAMON, KCMON and SUBRTN are not applicable. There are no sense switch options. All error reporting, looping, and so on, is handled within the controller program.

2.2.3 Operating systems (TOPS-10)

The Responder will run under TOPS-10 release 7.03 and newer, and uses the SCS. UUO and the DIAG. UUO.

2.2.4 Operating systems (TOPS-20)

The Responder will run under TOPS-20 release 6.0 and newer, and uses the SCS% JSYS and the DIAG% JSYS.

2.2.5 Spear

When running under TOPS-20, this program will be upward compatible with all versions of SPEAR. Since TOPS-10 lacks a method for a user program to create SPEAR entries, no SPEAR entries are created under TOPS-10.

2.3 Failsoft Goals

2.3.1 CPU/Memory Failures

The Responder will be aborted by the Operating System.

2.3.2 Power fail/restart

The Responder will be restarted when the System is reloaded.

2.4 Restrictions

There should only be one Responder running on a system at any one time. The Responder could be run continuously as an Operator job. However on TOPS-10 the responder has to lock itself in core so running it continuously is not a good idea. Under TOPS-20 the system could start it during boot time with no ill effects.

2.4.1 User Mode Privileges

Under TOPS-20, the SCS% JSYS requires Wheel, Operator or Maintenance privileges. Under TOPS-10, the SCS. UUO requires the job be [1,2] or have JP.POK privileges.

2.4.2 User Mode Priority Interrupts

The Responder does not use any Hardware Interrupts. The Software Interrupt System is used on both TOPS-10 and TOPS-20.

2.5 Non-Goals

- The Responder does not do error retries unless directed to do so by a Controller.
- There is no user interface.
- There is no exec mode support.

3.0 Requirements

3.1 Hardware Design Requirements

No special hardware features are required.

3.2 Run Time Requirements

- A TOPS-10 or TOPS-20 Timesharing Monitor needs to be running.
- Adequate user privileges.

3.2.1 CPU family

The KL10 CPU family with a CI20 port and Star Coupler.

3.2.2 Memory

Any configuration running the supported versions of the Operating Systems have sufficient memory.

3.2.3 Load Device

The load device will be the system disk.

3.2.4 User Interface

There is not any human interface to the Responder. All the human interfacing will take place at the controller program.

3.2.5 Operating Systems

TOPS-10 and TOPS-20 are the only supported Operating Systems.

TOPS-20 Operating System, release 6.0 or newer, supports necessary CI Diagnostic Features. TOPS-10 Operating System, release 7.03 or newer, supports the necessary CI Diagnostic Features.

4.0 Functional Description (Overview)

Under TOPS-20, the Responder is run on a priviledged OPERATOR job and can be loaded and started during system boot time. It can also be loaded and started by an Operator.

Under TOPS-10, the Responder should only be run when needed since it locks itself in core due to a restiction in TOPS-10's SCS UUO.

Once started, the Responder will execute an SCS LISTEN command which will allocate a Connection Block and listen for a connection request from a Controller node. The Monitor will interrupt the Responder when a Controller node wants a connection. The Responder will establish connections and send communications messages through the Operating System. The Responder uses the SCS% JSYS (TOPS-20) or SCS. UUO (TOPS-10) to send CTP Protocol messages across the CI. The Responder communicates with the Controller Process using CTP Protocol.

The Responder is able to talk to many Controllers at the same time. If by some reason the Responder stops running it will have to be restarted by the Operator.

4.1 Error Reporting Philosophy

- Report back to Controller on functional errors.
- Publish a message and stop if an impossible error condition exists. If the Responder fails in this manner a report should be made to Diagnostic Engineering.

4.1.1 Error Reporting of Hardware Detected Errors

Hardware errors are detected and logged by the Operating System.

4.1.2 Reporting to SPEAR

Under TOPS-20, the following information is logged into the System Error Log for SPEAR use.

- The number of connections from each node to the Responder since it was started.
- The number of connections from each node to the Responder since the last SPEAR entry.

No information is logged to SPEAR on TOPS-10, because TOPS-10 lacks the facility to create SPEAR entries from user mode programs.

4.2 Response Decoding

The required Response is determined by the opcode of the incoming CTP Requests (see Appendix B). The Responder is a slave to the Controller.

4.3 General Test Philosophy

Responders on the CI must respond to some subset of the cluster test protocol commands listed in the CTP document. Appendix B of this document lists in detail, all of the command, Response Messages that are supported in this program. Typical operation involves receiving a CTP request, interpreting it, taking the appropriate action, and returning a CTP response to the controller from which the request originated. The diagram in Appendix A Figure A shows an example of the use of CTP in a test that moves data from the responder to the controller. Note the pairing of the CTP request and response. In general, the responder always returns one or more CTP responses for each CTP request.

4.3.1 Responder Data Structure

The Connection Blocks are used to store information about Virtual Circuits (one Connection Block/Virtual Circuit). Only when a connection is accepted by the Responder is a Connection Block allocated to a Virtual Circuit.

Connection Block (CBLKn) Format:

Connect ID
Status
1ST RHBLK Address
2ND RHBLK Address
System Block Index,,Port Number
Local Port Number (CI port number)
1ST - SCA's Buffer Name
1ST - Our Buffer Name (BUFx) of above word
2ND - SCA's Buffer Name
2ND - Our Buffer Name (BUFx) of above word

The Request Header Blocks are used to store information about CTP requests (one Request Header Block/CTP Request). Only when a CTP request is received by the Responder is a Request Header Block allocated.

Request Header Block (RHn) Format:

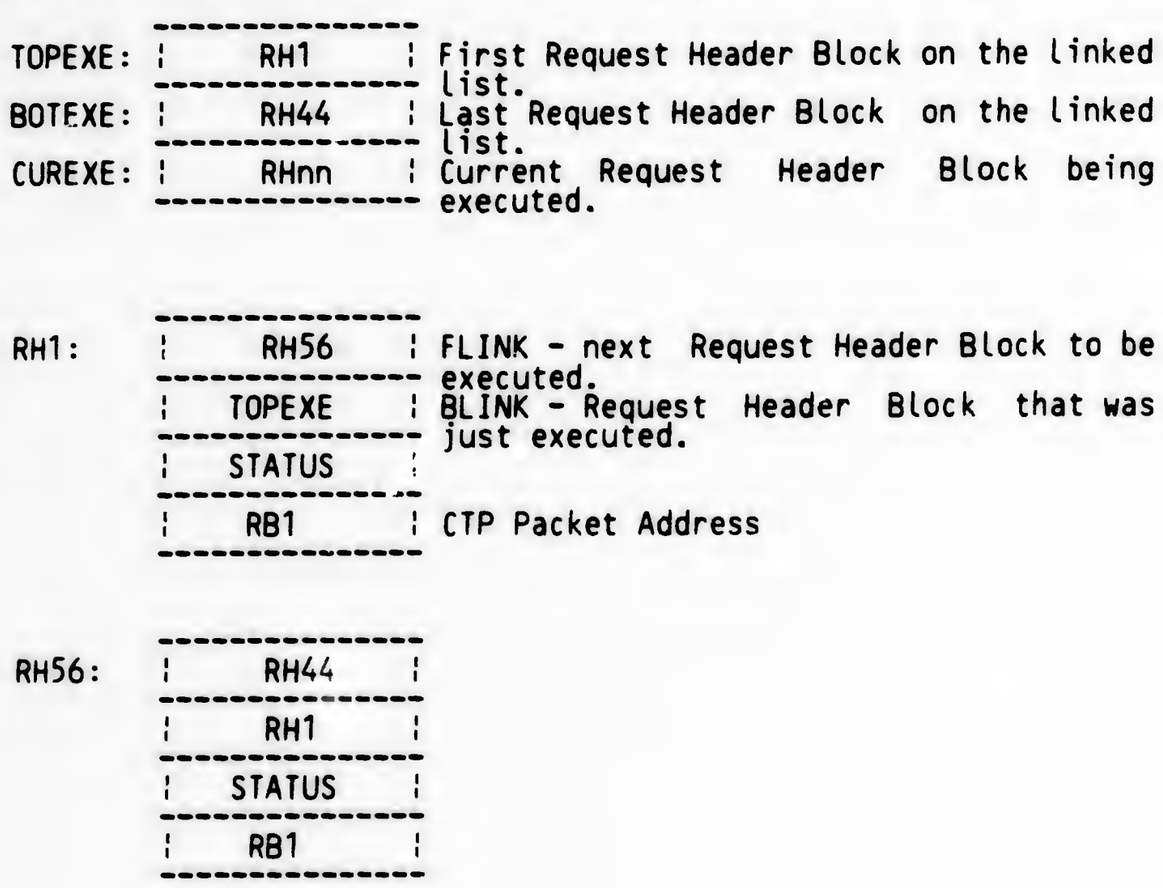
6 bit CID (3rd Party),,Forward Link (FLINK)
Repeat Count,,Backward Link (BLINK)
Status,,CBLKn Address
Start of CTP packet (RBn address)

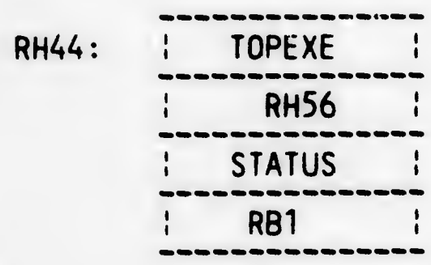
When a request has been received it must be executed. First, the Request Header Block is linked into the execution list as follows. The Responder has two pointers used to point to the top and bottom of the execution list. They keep track of the order that the request are received, and therefore the order in which they should be executed. If another request arrives from a Controller that already has a request on the linked list, it must wait for the first request to complete before the second is put on the linked list.

Once there is at least one CTP request to process, the mainline routine can begin executing. It will find the first Request Header Block by looking at the location "TOPEXE", which points to the first Request Header Block to execute. The value of topexe is written to "CUREXE". From now on CUREXE is used to keep track of which Request Header Block is currently being executed.

The mainline routine finds the CTP request, which is pointed to by a field in the Request Header Block, and cases off the CTP opcode to request specific routines to handle each of the possible CTP requests.

An example of how the FLINK and the BLINK words are used when there are 3 requests on the linked list awaiting execution:





4.3.2 Delay/Timeout Data Structure

If a request has a delay count, the request can not be executed immediately. A TIMER JSYS (TOPS-20) or PITMR UUO (TOPS-10) is executed which will signal, via the software interrupt system, when it is time to execute the request. The code executed at the timer interrrupt sets the "GOBIT" in the Request Header Block signifying that this CTP packet is now ready to be executed.

The data structure associated with this request is three parallel tables. The first table, DLYADR, contains the addresses of Request Header Blocks that are currently being timed. A zero in the table indicates that this slot is free. The second table, DLYUPT, contains the system uptime in milliseconds at which time the delay has been satisfied. The third table, DLYRTN, contains the address of a small routine to call to either set the GOBIT (requested delay count) or OVRDUE (to timeout a connection).

4.3.3 Third Party Data Structure

When a Controller wants the Responder to do a CTP Request that will include third party, he must first send a CTP Connect Request which will cause the Responder to open up a connection to the named Responder. When a connection is made the 6 bit cid that references that third party connection is placed into the left half of the FLINK word in the RHBLK.

```
Example:  ;This RHBLK is used to hold a third party CTP Request
          ;message.  It will get put on the Linked List.

          RHn:    6 bit cid,,FLINK
                  Repcount,,BLINK
                  Status,,CBLK
                  0,,Address of CTP request
```

The Responders' Interrupt Routine (CONACC - connection was accepted) will then get 2 unused RHBLKs and set them up so they will be able to be used when the response messages come in from the third party

Responder.

Example: ;These 2 RHBLKS will never be put on the Linked List.

```
RHn:    -1
        ?
        0,,CBLK
        0,,Address of CTP request

RHn:    -1
        ?
        0,,CBLK
        0,,Address of CTP request
```

The CBLK also gets set up:

```
CBLKn:  CID
        THIRD!GTINFO,,0
        ?
        ?
```

The Mainline Routine has a subroutine that gets called to set up the CBLKs with more information. All the CBLKs that have the "GTINFO" bit set in its status word will get updated in this routine.

```
Example:    CBLKn:  ?
                    GTINFO,,0               ;"GTINFO" gets cleared
                    ?
                    ?
                    SBI,,Node number
                    CI Port number
```

When we send out a CTP Request to a third party Responder, we will receive a CTP Response back from that Responder. We receive the Response from the interrupt routine "INPMSG". This routine can set up the CBLK as follows:

```
Example:    CBLKn:  ?
                    GA1CON!GA2CON!GANRES
                    RHBLK
                    RHBLK
                    ?
                    ?
```

4.3.4 Request execution scheme

The responder will often have many requests to service simultaneously, so it must have some mechanism for determining the order in which to respond to each request. The method used is a round robin scheme. For example, suppose two requests are received by the responder, each with a repeat count of three. The responder will execute the request received first once, then execute the other request, and repeat this cycle until both requests have been completed. This method ensures that requests with long repeat counts do not lock out the servicing of other waiting requests.

The exception to this rule occurs when two CTP requests originate from the same controller. In this case the responder executes the request received first until it has been completed before executing the second request.

An example of this algorithm is shown below. The second column in the diagram contains symbols representing the source of the CTP request. Also, the top of the list contains the request received first, the bottom contains the request received last.

REQUEST 1	A	REP CNT 2	SEND DG
REQUEST 2	B	REP CNT 3	SEND MSG
REQUEST 3	A	REP CNT 4	SEND MSG
REQUEST 4	B	REP CNT 3	SEND DATA

From the diagram above, four requests have been received, two each from node A and node B. REQUEST 3 will not be executed until REQUEST 1 is finished, and REQUEST 4 will not be executed until REQUEST 2 is finished.

Thus, the order of execution for the above diagram is as follows:

REQUEST 1
REQUEST 2
REQUEST 1
REQUEST 2
REQUEST 3
REQUEST 2
REQUEST 3
REQUEST 4
REQUEST 3
REQUEST 4
REQUEST 3
REQUEST 4

4.4 Command/Response Descriptions

The following sections provide verbal descriptions of all the CTP requests, supported by this Software, along with the appropriate Responses which must be taken for each. A detailed description of all the CTP Protocol implemented by the Responder is contained in Appendix B of this document.

4.4.1 Responder Function Set Request

This CTP request causes the responder to return a CTP function set response to the controller. The function set response contains a 32 byte field specifying the CTP functions implemented by the responder. Each bit in the 256 bit field corresponds to a CTP opcode which is implemented by the responder. If a bit is set, the responder implements the function specified by that request.

If a bit is clear, then the responder can receive that particular CTP request, but will return a response with a NOT IMPLEMENTED status field.

4.4.2 Buffer Map Request

This CTP message tells the responder to map a buffer in the responder's node for access by the controller. Three types of buffer are distinguished in a field of the incoming request. Real Memory buffer is mapped when the controller wishes to examine the contents of the buffer after a transfer has occurred. "Black Hole" buffer is used for test cases in which the controller wishes to create a high level of CI traffic but does not care to check the contents of the buffer after the transfer. The Maintenance Data Buffer is not used in this responder. The size of the desired buffer area is also given in a field in the buffer map request.

When mapped, buffers are filled with one of three types of data patterns specified by a field in the CTP request. They are: byte pairs (bytes zero and one contain 00, bytes two and three contains 01, etc); the responder's node number; or a constant given in another field in the request. This allows a buffer to be read without first writing a specific data pattern to it. The appropriate information needed by the controller to access the buffer including the buffer name and size is sent back to the controller in the CTP Buffer Map Response.

The buffer map information should be kept on a per-connection basis. The responder will unmap all the buffers for a connection if the connection is broken. Whenever the responder detects the breaking of a connection, it should check for and unmap any buffers mapped over that connection.

4.4.3 Buffer Unmap Request

This CTP message causes the responder to surrender a previously mapped buffer. The name of the buffer along with the buffer type and size are contained in fields in the request. When the unmapping has been completed, the responder returns a CTP Buffer Unmap Response to the controller.

4.4.4 Move Buffer Request

This CTP message causes the responder to move a block of data between itself and another node. Fields in the request specify the direction of the data movement, the other node involved, and the buffer names and offsets. A repeat count in the request makes it possible to perform multiple buffer transfers and a delay count in the request causes a delay (in 500ms increments) before starting the first transfer.

After the buffer transfer has been completed, the responder returns a CTP Move Buffer Response to the controller. This response contains a byte of status about the last move performed, a longword containing the Port status used for the last buffer transfer, along with the information contained in the original request.

A connection must exist between the two nodes prior to performing the buffer transfer. If no connection exists, the responder will return status back to the controller stating so.

It is not the responsibility of the responder, however, to map buffers that have not yet been mapped when the move buffer request is received.

4.4.5 Generate Message Request

This CTP message causes the responder to send the controller a stream of CTP Generate Message Response messages. Fields in the incoming request specify a repeat count, a delay (in 500ms increments) prior to returning the first response, the data with which to fill the response, and the length of the CTP response. The CTP response contains fields to reflect the status of the last message sent, a count field to indicate the count of the message sent, a field specifying the type of data contained in the response message, and the N bytes of generated data.

The number of messages sent by the responder should be one greater than the repeat count specified in the request. Thus, for a repeat count of zero, one response should be returned, and a repeat count of one should yield two responses.

4.4.6 Generate Datagram Request

This CTP message causes the responder to send the controller a stream of CTP Generate Datagram Response messages. Fields in the incoming request specify a repeat count, a delay (in 500ms increments) prior to returning the first response, the data with which to fill the response, and the length of the CTP response. The CTP response contains fields to reflect the status of the last datagram sent, a count field to indicate the count of the datagram sent, a field specifying the type of data contained in the response, and the N bytes of generated data.

The number of datagrams sent by the responder should be one greater than the repeat count specified in the request. Thus, for a repeat count of zero, one response should be returned, and a repeat count of one should yield two responses.

4.4.7 Generate Reset Request

This function will cause the responder to generate a specific reset packet on the CI. This request contains a target node and a extend field specifying whether to do a conditional or forced reset. This function will be used only when a CINT is connected to the CI.

4.4.8 Generate Start Request

This function will cause the responder to generate a specific start packet on the CI. This request contains a target node and a extend field specifying whether or not the start is at the node's default or the packet's specified address. Another field contains the node-specific start address. This function will be used only when a CINT is connected to the CI.

4.4.9 Start/Stop Unsolicited Activity

This CTP message causes the responding system to either stop or resume activity on the CI. This means that if a node has a poller that checks configuration information by executing request ID's to other nodes, then upon receipt of this request this activity is stopped. This function will be used only when a CINT is connected to the CI.

4.4.10 Configuration Data Request

This CTP message causes the responder to collect from its own (or SCA's) internal data tables information about a third node on the CI. The responder returns this information to the controller in a CTP Configuration Data Response Message. Included in this response are fields containing information about the target node's:

- Number of node
- Path A status
- Path B status

A status field is also contained in the configuration response. The responder may indicate a refusal to gather the information, may not have had knowledge about the target node, or may indicate a successful gathering of the information in this field.

4.4.11 Counter Read Request

This CTP message causes the responder to read the event counters maintained in its own CI port. The responder then returns a CTP Counter Read Response which contains the contents of the eight counters (ACK, NAK, NORSP and DISC for each path) and the status of the counter reads.

4.4.12 Connect Request

For third party buffer transfers it is necessary for connections to be established between responders. This is accomplished by using this function.

4.4.13 Listener Disconnect Request

This CTP message causes the responder to shut down its communications with the controller in an orderly manner. Upon receiving this CTP request, the responder deallocates any resources used for the connection. It then returns a CTP Listener Disconnect Response to the controller with a status field indicating a successful orderly disconnection. After sending this CTP response, the responder issues a local SCA disconnect request.

5.0 Program Interfaces

5.1 Diagnostic Software Interfaces

Not Applicable.

5.2 Operating System Interfaces (TOPS-10)

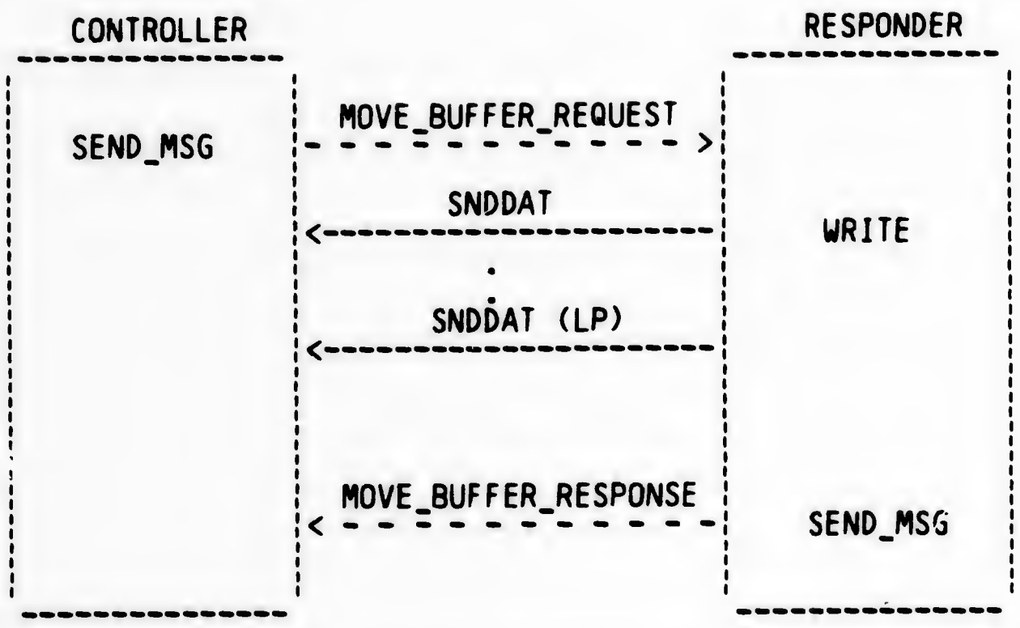
The programming interface is with the monitor using the SCS. UUO, as implemented in TOPS-10 release 7.03 and later.

5.3 Operating System Interfaces (TOPS-20)

The programming interface is with the monitor. Packets of information will be sent and received using the SCS% JSYS, supported in TOPS-20 Release 6.0 and later.

6.0 Appendices

6.1 Appendix A (Figures)



- - - - = Cluster Test Protocol Request/Response

------- = CI Data Link Packet

Figure A

6.2 Appendix B (CTP Messages)

6.2.1 CTP Message Formats

This section describes the formats of the messages that are passed between processes speaking CTP. All of the messages are sent using the SCA, and this section does not address the SCA portions of the messages. Refer to the SCA documents for those formats.

This section merely presents the encoding formats and syntax of the messages used in CTP version 2.

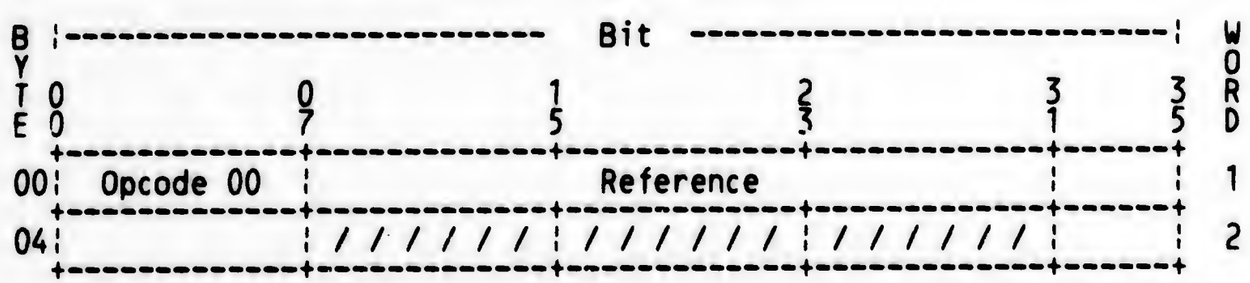
Note that all messages begin with a one-byte opcode, indicating what type the message is. Digital reserves opcodes in the range 0 to 127, while values in the range 128 to 254 are reserved for customer and CSS additions to the protocol. Opcodes generated by the responder as a result of a message from the controller will be 64 greater than the controller's opcode. The opcode value 255 is reserved for extending the opcode space to multiple byte values.

All messages have a four-byte reference number in the second through fifth bytes. This number is not used by the responder.

All responses contain a status code in the sixth byte. Various possibilities for success, failure and refusal can be described by the responder in the status field. In all cases, successful status values are indicated using positive values in the status field, while negative status field values indicate a failure or refusal on the part of the responder.

6.2.2 Responder Function Set Messages

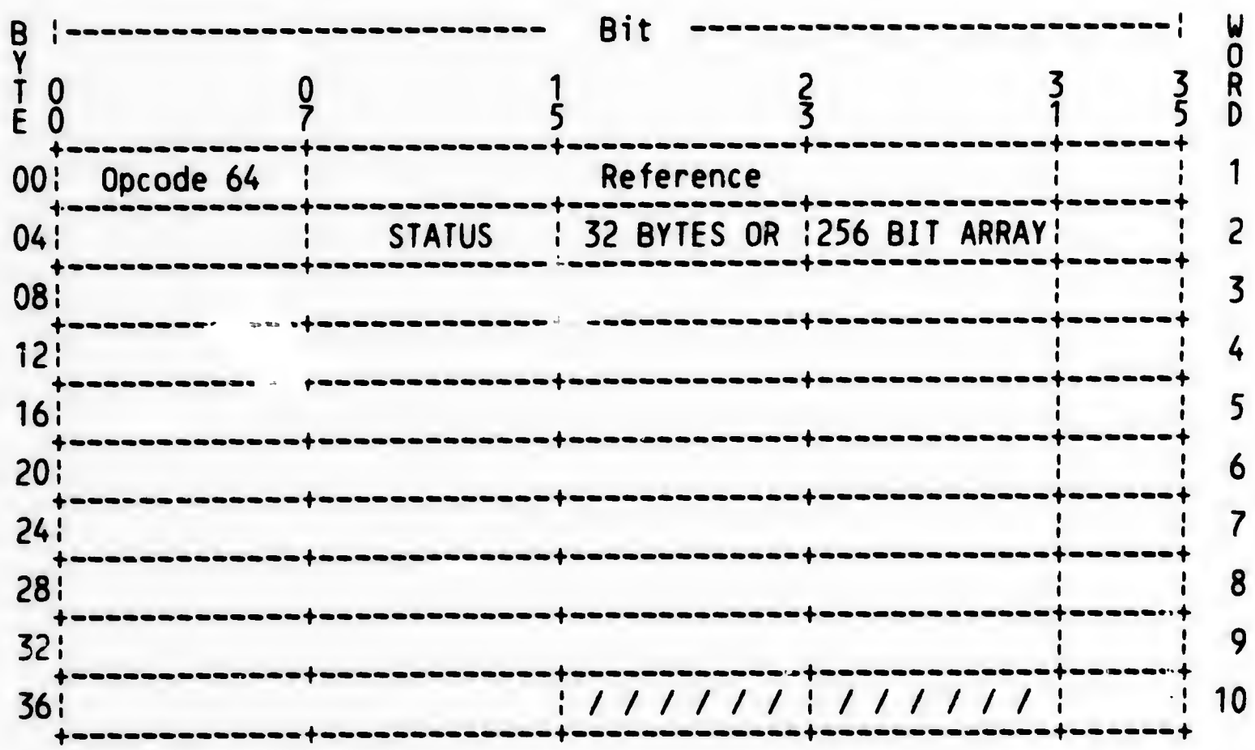
Responder Function Set Request Message



00/CTP$OPCODE 1 Byte, value is 00/CTP$FUNCTREQ.

01/CTP$REFERENCE 4 Bytes, contains reference number.

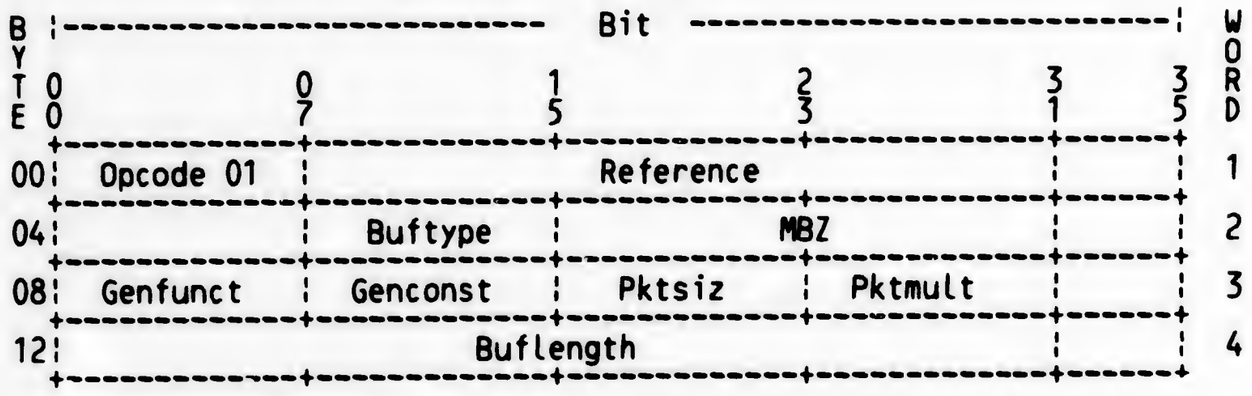
Responder Function Set Response Message



00/CTP$OPCODE	1 Byte, value is 64/CTP$FUNCTRSP.
01/CTP$REFERENCE	4 Bytes, contains reference number.
05/CTP$STATUS	1 Byte, value is 00/CTP$SUCCESS.
06/CTP$FMASK	32 Bytes, arranged as a 256 bit array, with each bit set if the opcode is implemented by the responder and cleared otherwise. The least significant bit of the first byte corresponds to opcode zero, and the most significant bit of the 32nd byte corresponds to opcode 255. For CTP functions not implemented, the responder must still be able to receive the CTP packet type, but should return the response with a NOT IMPLEMENTED status value.

6.2.3 Buffer Map Messages

Buffer Map Request Message



00/CTP$OPCODE — 1 Byte, value is 01/CTP$BUFMAPREQ.

01/CTP$REFERENCE — 4 Bytes, contains reference number.

05/CTP$BUFTYPE — 1 Byte indicating type of buffer to allocate, values are:

- 01/CTP$REALBUF — Real Memory Buffer.
- 02/CTP$FAKEBUF — "Black Hole" Buffer.

06/CTP$RESERV06 — 2 Bytes, must be zero.

08/CTP$GENFUNCT — 1 Byte specifying the function to use for the initial contents of the allocated buffer. Values are:

- 00/CTP$GENFFILL — Fill buffer with constant.
- 01/CTP$GENFBPAIR — Fill w/byte pair numbers.
- 02/CTP$GENFNODE — Fill w/responder node number

09/CTP$GENCONST — 1 Byte holding the fill constant, only used with the 00/CTP$GENFFILL function.

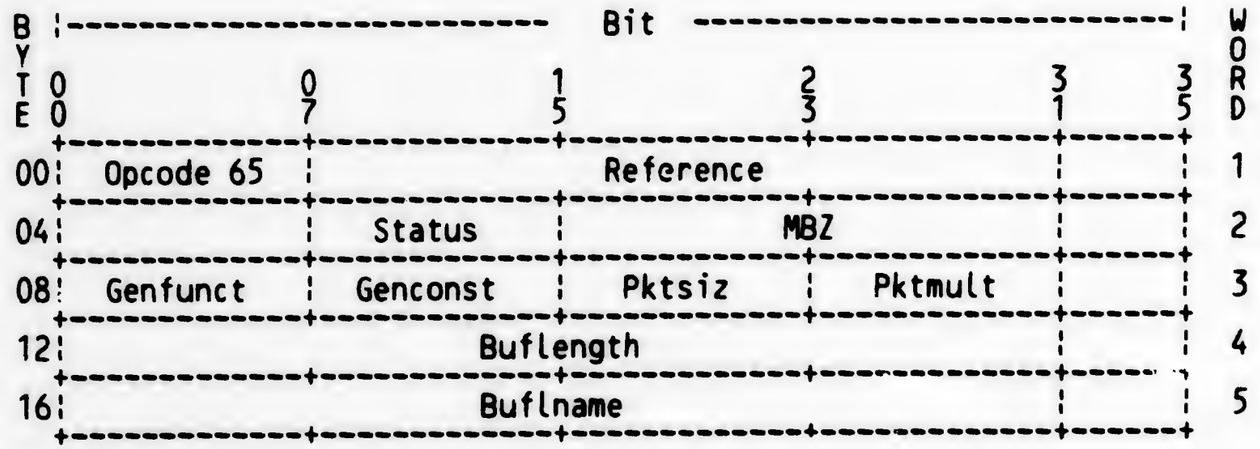
10/CTP$PKTSIZ — 1 Byte, contains the base packet size to use for the transfer.

- 00/CTP$PS512 — Size is a multiple of 512 bytes.
- 01/CTP$PS576 — Size is a multiple of 576 bytes.

11/CTP$PKTMULT — 1 Byte, specifying the packet size multiple.

12/CTP$BUFLENGTH — 4 Byte integer indicating number of bytes to allocate.

Buffer Map Response Message



00/CTP$OPCODE 1 Byte, value is 65/CTP$BUFMAPRSP.

01/CTP$REFERENCE 4 Bytes, contains reference number.

05/CTP$STATUS 1 Byte indicating type of buffer allocated, values are:

001.(001)/CTP$REALBUF Real Memory Buffer.
002.(002)/CTP$FAKEBUF "Black Hole" Buffer.
250.(372)/CTP$BUFLIMIT Buffer Map Limit Exceeded.
253.(375)/CTP$NORESOURCE Resource Failure.
254.(376)/CTP$NONSENSE Request Not Understood.
255.(377)/CTP$NOTIMP Function Not Implemented.

06/CTP$RESERV06 2 Bytes, must be zero.

08/CTP$GENFUNCT 1 Byte specifying the function to use for the initial contents of the allocated buffer. Values are:

00/CTP$GENFFILL Fill buffer with constant.
01/CTP$GENFBPAIR Fill w/byte pair numbers.
02/CTP$GENFNODE Fill w/responder node number

09/CTP$GENCONST 1 Byte holding the fill constant, only used with the 00/CTP$GENFFILL function.

10/CTP$PKTSIZ 1 Byte, contains the base packet size to use for the transfer.

00/CTP$PS512 Size is a multiple of 512 bytes.
01/CTP$PS576 Size is a multiple of 576 bytes.

11/CTP$PKTMULT 1 Byte, specifying the packet size multiple.

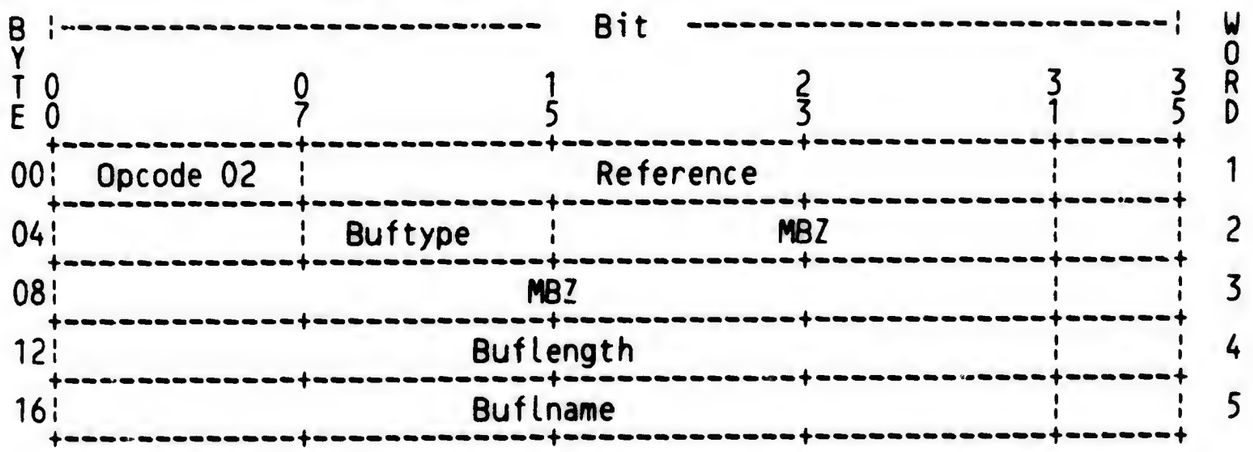
12/CTP$BUFLENGTH 4 Byte integer with number of bytes

actually allocated.

16/CTP$BUFLNAME	4 Byte value with the name of the buffer allocated.

6.2.4 Buffer Unmap Messages

Buffer Unmap Request Message



00/CTP$OPCODE — 1 Byte, value is 02/CTP$BUFUNMREQ.

01/CTP$REFERENCE — 4 Bytes, contains reference number.

05/CTP$BUFTYPE — 1 Byte indicating type of buffer to deallocate, values:

- 01/CTP$REALBUF — Real Memory Buffer.
- 02/CTP$FAKEBUF — "Black Hole" Buffer.

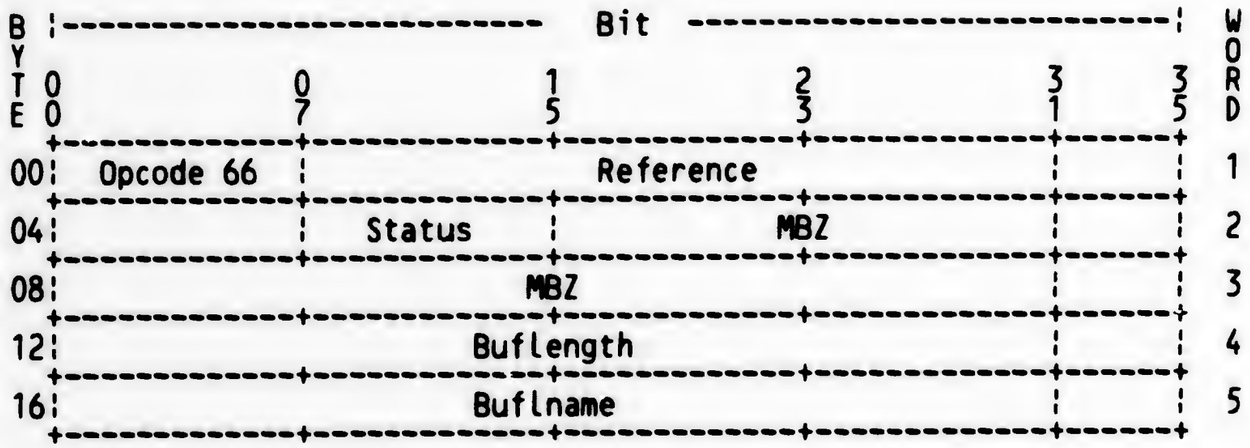
06/CTP$RESERV6 — 4 Bytes, must be zero.

10/CTP$RESERV10 — 2 Bytes, must be zero.

12/CTP$BUFLENGTH — 4 Byte integer indicating number of bytes to release.

16/CTP$BUFLNAME — 4 Byte value with the name of the buffer to release.

Buffer Unmap Response Message



00/CTP$OPCODE 1 Byte, value is 66/CTP$BUFUNMRSP.

01/CTP$REFERENCE 4 Bytes, contains reference number.

05/CTP$STATUS 1 Byte indicating type of buffer deallocated, values are:

001.(001)/CTP$REALBUF Real Memory Buffer.
002.(002)/CTP$FAKEBUF "Black Hole" Buffer.
249.(371)/CTP$NOBUFMAPPED No Buffer Mapped.
253.(375)/CTP$NORESOURCE Resource Failure.
254.(376)/CTP$NONSENSE Request Not Understood.
255.(377)/CTP$NOTIMP Function Not Implemented.

06/CTP$RESERV6 4 Bytes, must be zero.

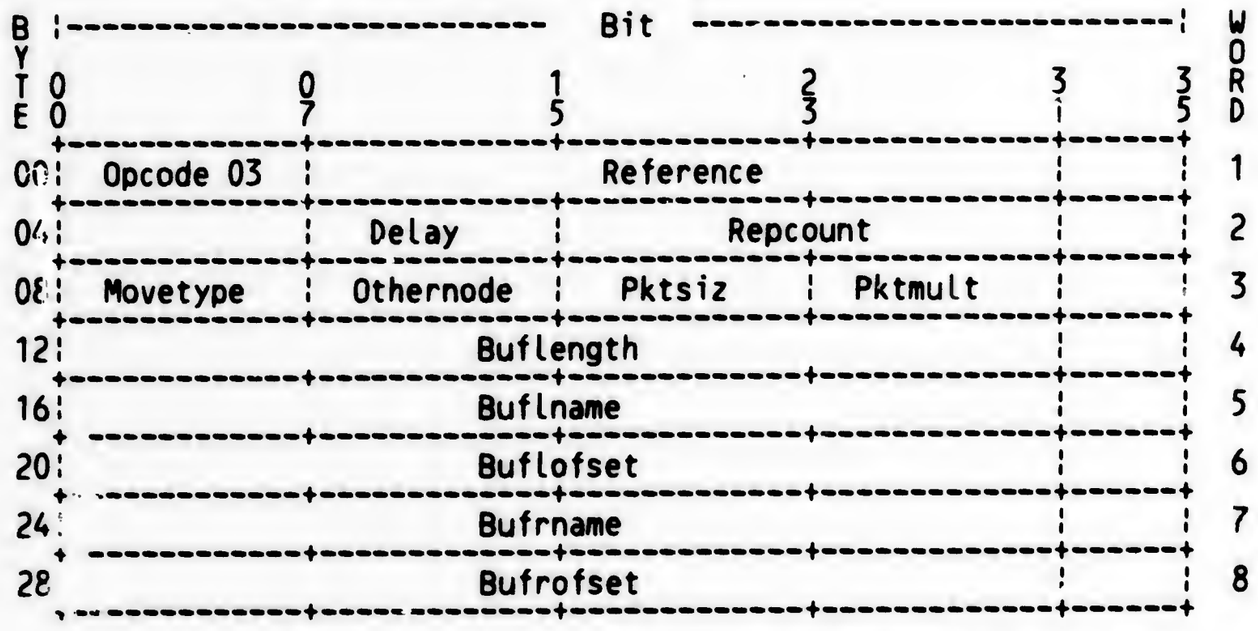
10/CTP$RESERV10 2 Bytes, must be zero.

12/CTP$BUFLENGTH 4 Byte integer indicating number of bytes released.

16/CTP$BUFLNAME 4 Byte value with the name of the buffer released.

6.2.5 Move Buffer Messages

Move Buffer Request Message



00/CTP$OPCODE 1 Byte, value is 03/CTP$MOVBUFREQ.

01/CTP$REFERENCE 4 Bytes, contains reference number.

05/CTP$DELAY 1 Byte specifying delay prior to action (~500ms/count).

06/CTP$REPCOUNT 2 Byte integer containing a repeat count.

08/CTP$MOVETYPE 1 Byte specifying the type of buffer move to perform:

- 00/CTP$MOVEFROM Responder does a read.
- 01/CTP$MOVETO Responder does a write.

09/CTP$OTHERNODE 1 Byte containing the number of the other node involved.

10/CTP$PKTSIZ 1 Byte, contains the base packet size to use for the transfer.

- 00/CTP$PS512 Size is a multiple of 512 bytes.
- 01/CTP$PS576 Size is a multiple of 576 bytes.

11/CTP$PKTMULT 1 Byte, specifying the packet size multiple.

12/CTP$BUFLENGTH 4 Byte integer containing the size of the move in bytes.

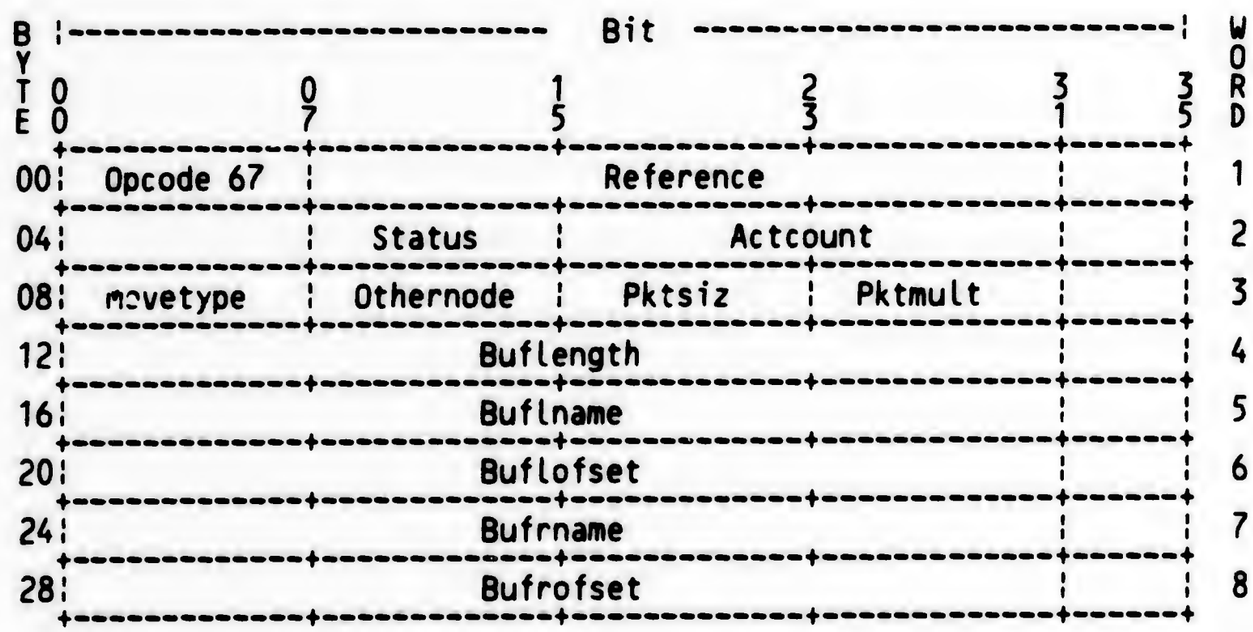
16/CTP$BUFLNAME 4 Byte value with the local buffer name.

20/CTP$BUFLOFSET 4 Byte value containing the local buffer offset.

24/CTP$BUFRNAME 4 Byte value with the remote buffer name.

28/CTP$BUFROFSET 4 Byte value containing the remote buffer offset.

Move Buffer Response Message



00/CTP$OPCODE 1 Byte, value is 67/CTP$MOVBUFRSP.

01/CTP$REFERENCE 4 Bytes, contains reference number.

05/CTP$STATUS 1 Byte with the SCA or CTP status of the last move performed. Values are:

000.(000)/CTP$SUCCESS Function Successful.
246.(366)/CTP$NOCONNECT Noconnection exits.
247.(367)/CTP$PKTSIZFAIL Unable to use specified packet size
249.(371)/CTP$NOBUFMAPPED No Buffer Mapped.
251.(373)/CTP$FAILURE Function failure.
252.(374)/CTP$REFUSE Request refused.

253.(375)/CTP$NORESOURCE Resource Failure.
254.(376)/CTP$NONSENSE Request Not Understood.
255/CTP$NOTIMP Function Not Implemented.

06/CTP$ACTCOUNT 2 Byte integer holding the actual number of moves done.

08/CTP$MOVETYPE 1 Byte specifying the type of buffer move performed:

00/CTP$MOVEFROM Responder did a read.
01/CTP$MOVETO Responder did a write.

09/CTP$OTHERNODE 1 Byte containing the number of the other node involved.

10/CTP$PKTSIZ 1 Byte, contains the base packet size to use for the transfer.

00/CTP$PS512 Size is a multiple of 512 bytes.
01/CTP$PS576 Size is a multiple of 576 bytes.

11/CTP$PKTMULT 1 Byte, specifying the packet size multiple.

12/CTP$BUFLENGTH 4 Byte integer containing the size of the move in bytes.

16/CTP$BUFLNAME 4 Byte value with the local buffer name.

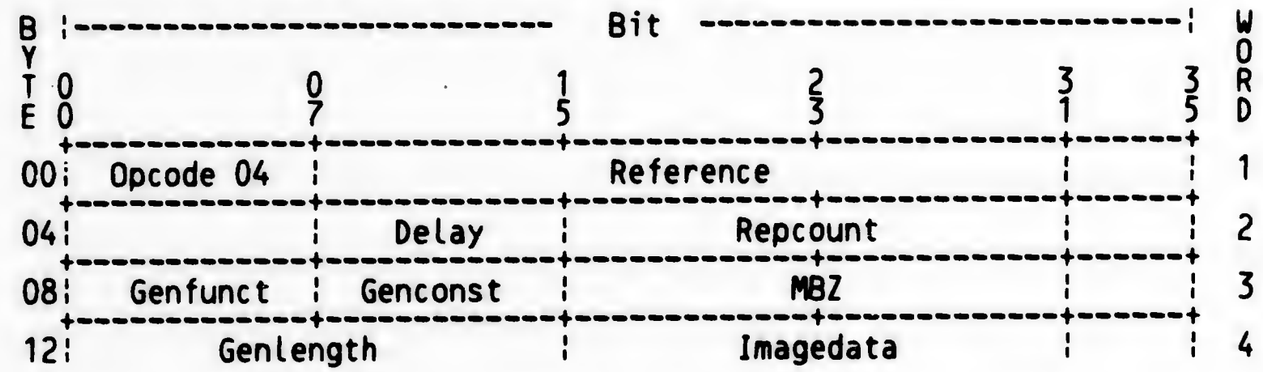
20/CTP$BUFLOFSET 4 Byte value containing the local buffer offset.

24/CTP$BUFRNAME 4 Byte value with the remote buffer name.

28/CTP$BUFROFSET 4 Byte value containing the remote buffer offset.

6.2.6 Generate Message Messages

Generate Message Request Message


Bit
BYTE
WORD
0 0 0 7 1 5 2 3 3 1 3 5
00 Opcode 04 Reference 1
04 Delay Repcount 2
08 Genfunct Genconst MBZ 3
12 Genlength Imagedata 4

00/CTP$OPCODE — 1 Byte, value is 04/CTP$GENMSGREQ.

01/CTP$REFERENCE — 4 Bytes, contains reference number.

05/CTP$DELAY — 1 Byte specifying delay prior to action (~500ms/count).

06/CTP$REPCOUNT — 2 Byte integer containing a repeat count.

08/CTP$GENFUNCT — 1 Byte specifying the function to use for the data field in the generated packets. Values are:

- 00/CTP$GENFFILL — Fill packet with constant.
- 01/CTP$GENFBPAIR — Fill w/byte pair numbers.
- 02/CTP$GENFNODE — Fill w/responder node number
- 03/CTP$GENFIMAGE — Fill w/data in this packet.

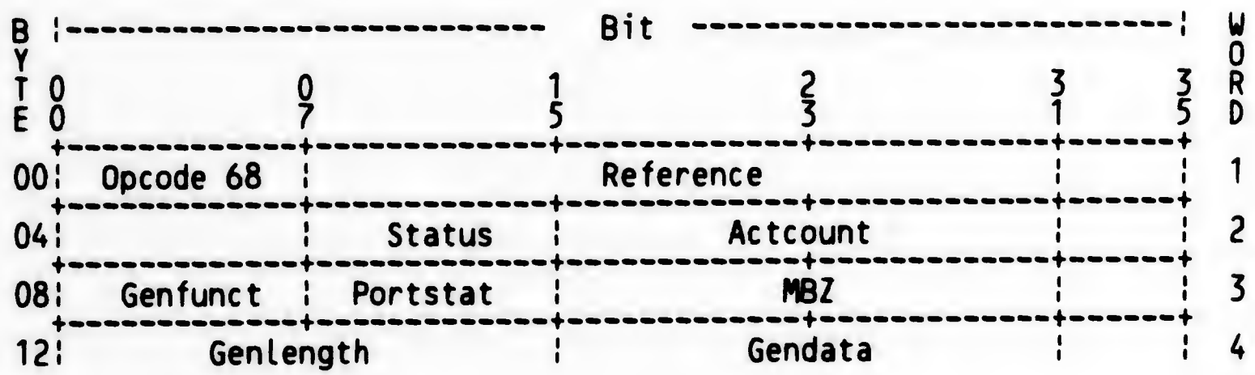
09/CTP$GENCONST — 1 Byte holding a fill constant if CTP$GENFFILL, or zero otherwise.

10/CTP$RESERV10 — 2 Bytes must be zero.

12/CTP$GENLENGTH — 2 Byte integer indicating the generated message length.

14/CTP$IMAGEDATA — N bytes of image function data if appropriate.

Generate Message Response Message



00/CTP$OPCODE 1 Byte, value is 68/CTP$GENMSGRSP.

01/CTP$REFERENCE 4 Bytes, contains reference number.

05/CTP$STATUS 1 Byte with the SCA or CTP status of the last message sent. Values are:

000.(000)/CTP$SUCCESS Function Successful.
253.(375)/CTP$NORESOURCE Resource Failure.
254.(376)/CTP$NONSENSE Request Not Understood.

06/CTP$ACTCOUNT 2 Byte integer containing the count of messages sent.

08/CTP$GENFUNCT 1 Byte specifying the function used for the data field in the generated packets. Values are:

00/CTP$GENFFILL Fill packet with constant.
01/CTP$GENFBPAIR Fill w/byte pair numbers.
02/CTP$GENFNODE Fill w/responder node number
03/CTP$GENFIMAGE Fill w/data in this packet.

09/CTP$PORTSTAT 1 Byte containing the port returned status of the last message sent.

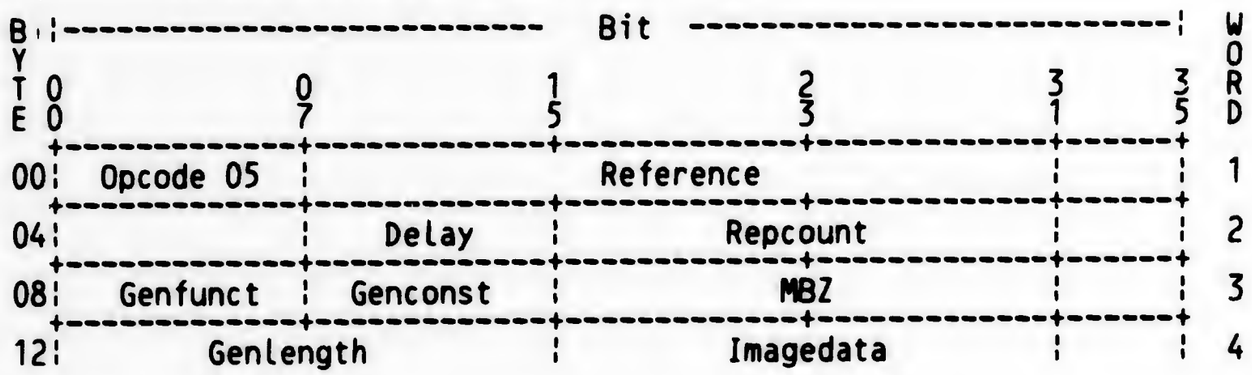
10/CTP$RESERV10 2 Bytes of filler, MBZ.

12/CTP$GENLENGTH 2 Byte integer indicating the generated message length.

14/CTP$GENDATA N Bytes containing the generated data.

6.2.7 Generate Datagram Messages

Generate Datagram Request Message



00/CTP$OPCODE — 1 Byte, value is 05/CTP$GENDGRREQ.

01/CTP$REFERENCE — 4 Bytes, contains reference number.

05/CTP$DELAY — 1 Byte specifying delay prior to action (~500ms/count).

06/CTP$REPCOUNT — 2 Byte integer containing a repeat count.

08/CTP$GENFUNCT — 1 Byte specifying the function to use for the data field in the generated packets. Values are:

- 00/CTP$GENFFILL — Fill packet with constant.
- 01/CTP$GENFBPAIR — Fill w/byte pair numbers.
- 02/CTP$GENFNODE — Fill w/responder node number
- 03/CTP$GENFIMAGE — Fill w/data in this packet.

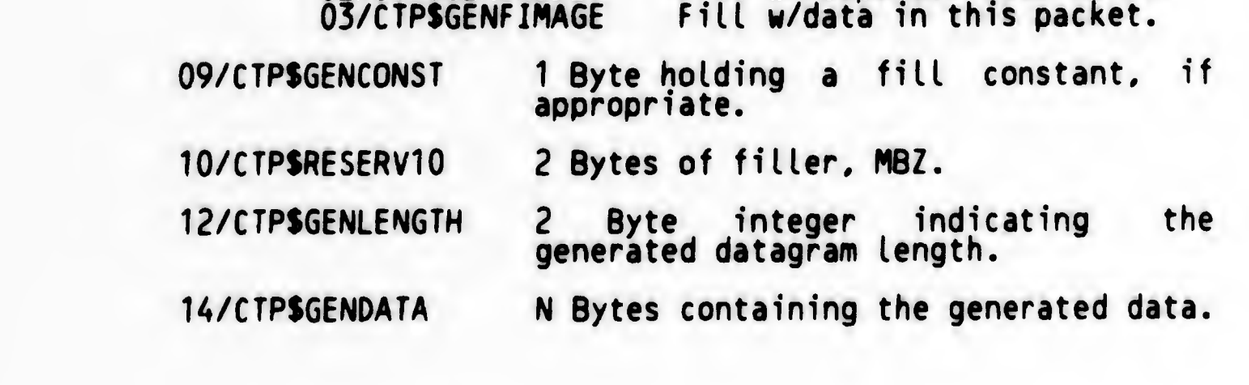
09/CTP$GENCONST — 1 Byte holding a fill constant, if appropriate.

10/CTP$RESERV10 — 2 Bytes must be zero.

12/CTP$GENLENGTH — 2 Byte integer indicating the generated DATAGRAM length.

14/CTP$IMAGEDATA — N bytes of image function data if appropriate.

Generate Datagram Response Message



00/CTP$OPCODE 1 Byte, value is 69/CTP$GENDGRRSP.

01/CTP$REFERENCE 4 Bytes, contains reference number.

05/CTP$STATUS 1 Byte with the SCA or CTP status of the last datagram sent. Values are:

000.(000)/CTP$SUCCESS Function Successful.
253.(375)/CTP$NORESOURCE Resource Failure.
254.(376)/CTP$NONSENSE Request Not Understood.

06/CTP$ACTCOUNT 2 Byte integer containing the count of datagrams sent.

08/CTP$GENFUNCT 1 Byte specifying the function to use for the data field in the generated packets. Values are:

00/CTP$GENFFILL Fill packet with constant.
01/CTP$GENFBPAIR Fill w/byte pair numbers.
02/CTP$GENFNODE Fill w/responder node number
03/CTP$GENFIMAGE Fill w/data in this packet.

09/CTP$GENCONST 1 Byte holding a fill constant, if appropriate.

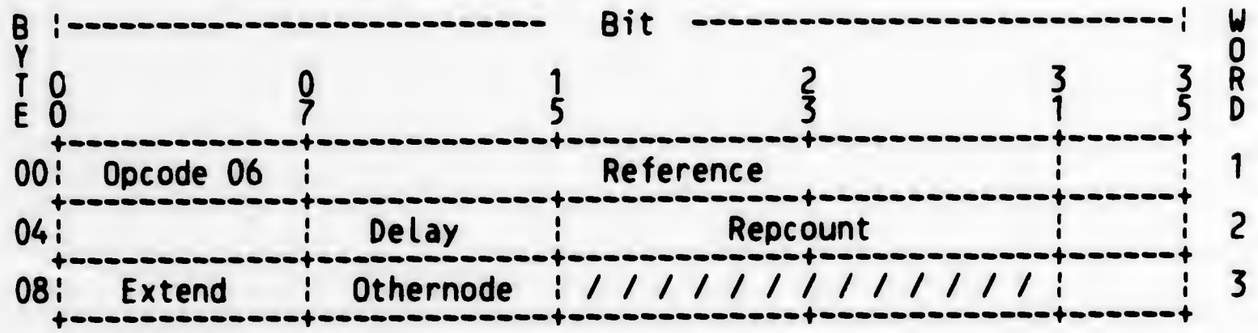
10/CTP$RESERV10 2 Bytes of filler, MBZ.

12/CTP$GENLENGTH 2 Byte integer indicating the generated datagram length.

14/CTP$GENDATA N Bytes containing the generated data.

6.2.8 Generate Reset Messages

Generate Reset Request Message



00/CTP$OPCODE — 1 Byte, value is 06/CTP$GENRSTREQ.

01/CTP$REFERENCE — 4 Bytes, contains reference number.

05/CTP$DELAY — 1 Byte specifying delay prior to action (~500ms/count).

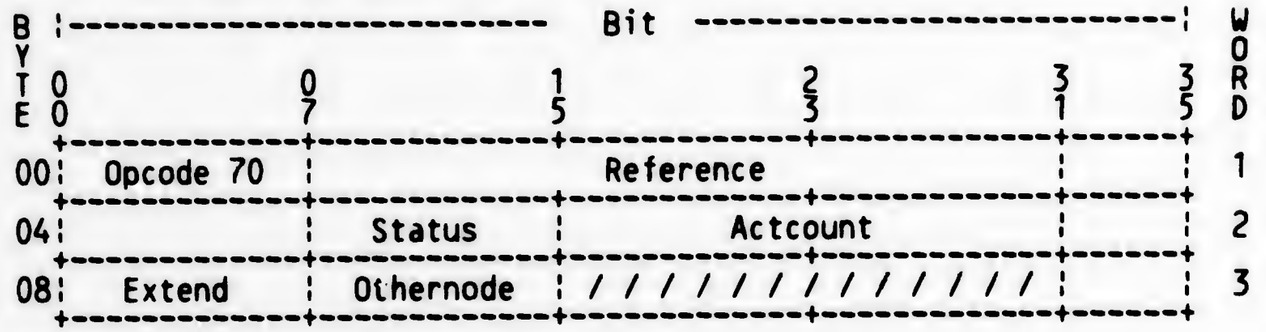
06/CTP$REPCOUNT — 2 Byte integer containing a repeat count.

08/CTP$EXTEND — 1 Bytes specifying whether the reset is conditional:

- 00/CTP$UNCONDRST — Forced reset.
- 01/CTP$CONDRST — Conditionally reset.

09/CTP$OTHERNODE — 1 Byte containing number of the node to reset.

Generate Reset Response Message



00/CTP$OPCODE 1 Byte, value is 70/CTP$GENRSTRSP.

01/CTP$REFERENCE 4 Bytes, contains reference number.

05/CTP$STATUS 1 Byte with the SCA or CTP status of the last reset request. Values are:

000.(000)/CTP$SUCCESS	Function Successful.
251.(373)/CTP$FAILURE	Function failure.
252.(374)/CTP$REFUSE	Request refused.
253.(375)/CTP$NORESOURCE	Resource Failure.
254.(376)/CTP$NONSENSE	Request Not Understood.
255.(377)/CTP$NOTIMP	Function Not Implemented.

06/CTP$ACTCOUNT 2 Byte integer containing number of resets sent.

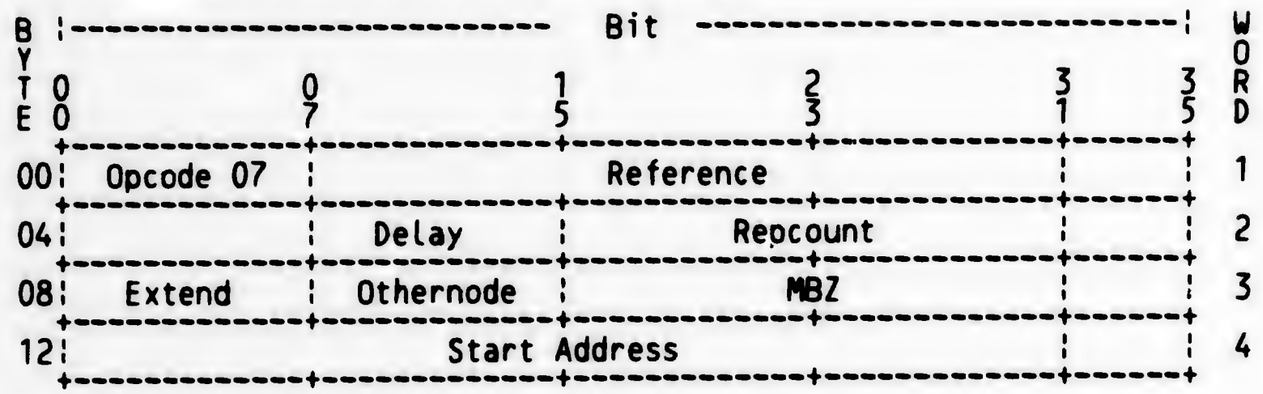
08/CTP$EXTEND 1 Bytes specifying whether the reset is conditional:

00/CTP$UNCONDRST	Forced reset.
01/CTP$CONDRST	Conditionally reset.

09/CTP$OTHERNODE 1 Byte containing number of the node that was reset.

6.2.9 Generate Start Messages

Generate Start Request Message



00/CTP$OPCODE	1 Byte, value is 07/CTP$GENSTRREQ.
01/CTP$REFERENCE	4 Bytes, contains reference number.
05/CTP$DELAY	1 Byte specifying delay prior to action (~500ms/count).
06/CTP$REPCOUNT	2 Byte integer containing a repeat count.
08/CTP$EXTEND	1 Byte specifying whether the start address is default:
00/CTP$DEFALTADR	Use node's default address.
01/CTP$SPECIFADR	Use the specified address.
09/CTP$OTHERNODE	1 Byte containing number of the node to start.
10/CTP$RESERV10	2 Bytes must be zero.
12/CTP$STARTADR	4 Byte value containing a node-specific start address.

Generate Start Response Message

BYTE	Bit 0-7	Bit 0-15	Bit 32-63	Bit 31-35	WORD
00	Opcode 71	Reference			1
04		Status	Actcount		2
08	Extend	Othernode	MBZ		3
12	Start Address				4

00/CTP$OPCODE 1 Byte, value is 71/CTP$GENSTRRSP.

01/CTP$REFERENCE 4 Bytes, contains reference number.

05/CTP$STATUS 1 Byte with the SCA or CTP status of the last start sent. Values are:

000.(000)/CTP$SUCCESS	Function Successful.
251.(373)/CTP$FAILURE	Function failure.
252.(374)/CTP$REFUSE	Request refused.
253.(375)/CTP$NORESOURCE	Resource Failure.
254.(376)/CTP$NONSENSE	Request Not Understood.
255.(377)/CTP$NOTIMP	Function Not Implemented.

06/CTP$ACTCOUNT 2 Byte integer containing number of starts sent.

08/CTP$EXTEND 1 Bytes specifying whether start address was defaulted:

00/CTP$DEFALTADR	Node's default was used.
01/CTP$SPECIFADR	Specified address was used.

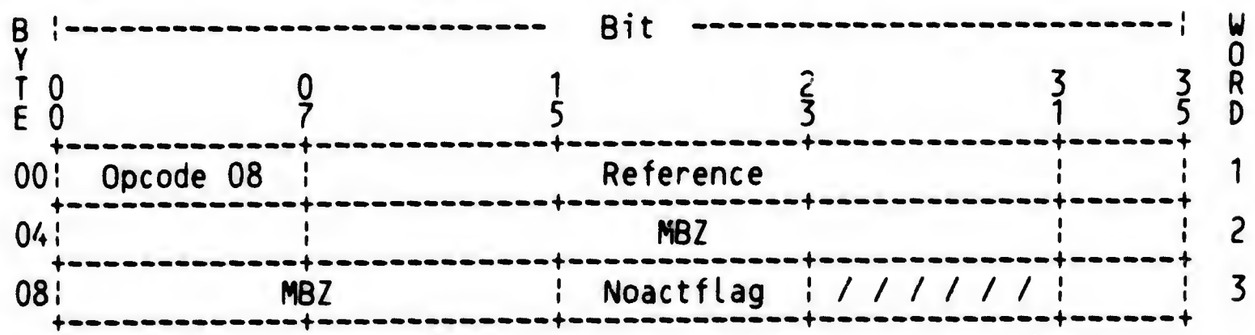
09/CTP$OTHERNODE 1 Byte containing number of the node started.

10/CTP$RESERV10 2 Bytes, must be zero.

12/CTP$STARTADR 4 Byte value containing the node-specific start address.

6.2.10 Start/Stop Unsolicited Activity Messages

Start/Stop Unsolicited Activity Request Message



00/CTP$OPCODE 1 Byte, value is 08/CTP$NOACTRSP.

01/CTP$REFERENCE 4 Bytes, contains reference number.

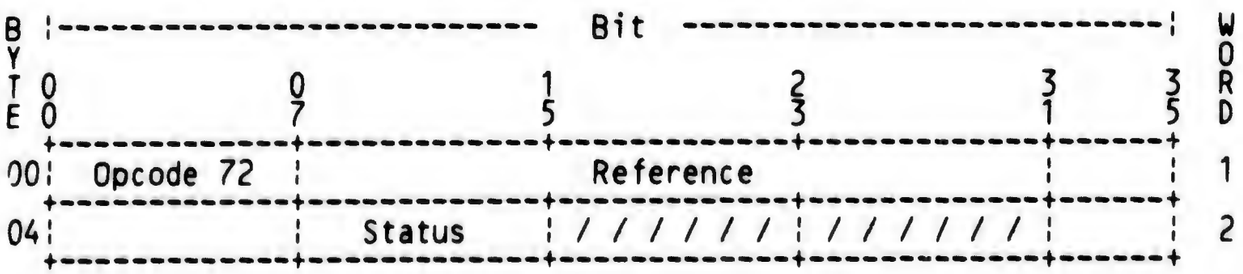
05/CTP$RESERV5 5 Bytes, must be zero.

10/CTP$NOACTFLAG 1 Byte contains flag to either stop or start normal unsolicited activity.

00/CTP$NOACTON Turn normal unsolicited activity on.
01/CTP$NOACTOFF Turn normal unsolicited activity off.

WARNING: If the poller has been stopped and the connection to the controller that requested the stop is lost, it is the responsibility of the responder to automatically restart the poller.

Start/Stop Unsolicited Activity Response Message



00/CTP$OPCODE 1 Byte, value is 72/CTP$NOACTRSP.

01/CTP$REFERENCE 4 Bytes, contains reference number.

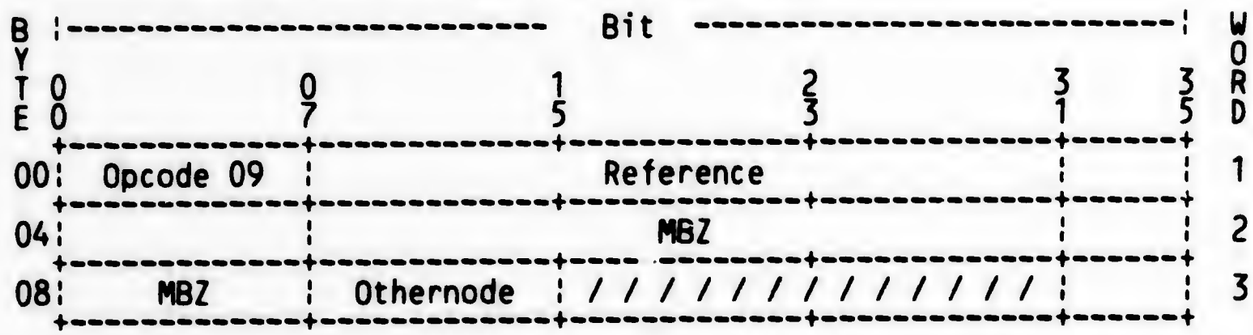
05/CTP$STATUS 1 Byte indicating status of request:

000.(000)/CTP$SUCCESS Function successful.
251.(373)/CTP$FAILURE Function failure.

252.(374)/CTP$REFUSE Request refused.
254.(376)/CTP$NONSENSE Request Not Understood.
255.(377)/CTP$NOTIMP Function Not Implemented.

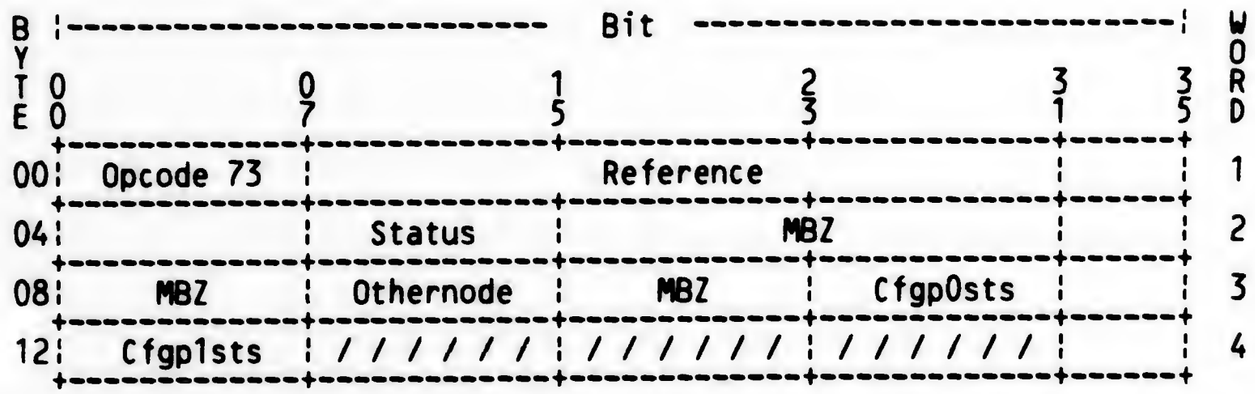
6.2.11 Configuration Data Messages

Configuration Data Request Message



00/CTP$OPCODE	1 Byte, value is 09/CTP$CONFIGREQ.
01/CTP$REFERENCE	4 Bytes, contains reference number.
05/CTP$RESERV5	5 Bytes, must be zero.
09/CTP$OTHERNODE	1 Byte containing number of node for which configuration information is desired.

Configuration Data Response Message



00/CTP$OPCODE 1 Byte, value is 73/CTP$CONFIGRSP.

01/CTP$REFERENCE 4 Bytes, contains reference number.

05/CTP$STATUS 1 Byte indicating knowledge of specified node:

000.(000)/CTP$SUCCESS Function successful.
248.(370)/CTP$NODEUNKNOWN Node Unknown To SCA.
253.(375)/CTP$NORESOURCE Resource Failure.
254.(376)/CTP$NONSENSE Request Not Understood.

06/CTP$RESERV6 3 Bytes, must be zero.

09/CTP$OTHERNODE 1 Byte containing number of node for information in this message.

11/CTP$CFGPOSTS 1 Byte containing path 0 status.

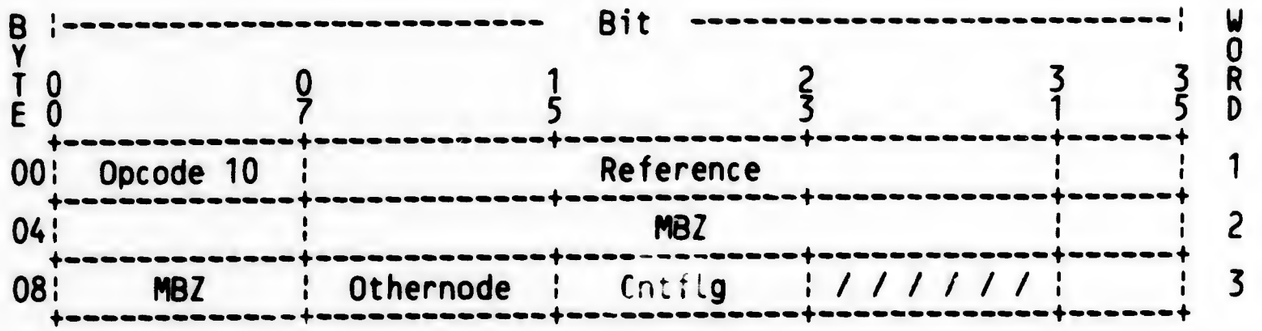
00/CTP$PATHBAD Path is marked bad.
01/CTP$PATHGOOD Path is marked good.

12/CTP$CFGP1STS 1 Byte containing path 1 status.

00/CTP$PATHBAD Path is marked bad.
01/CTP$PATHGOOD Path is marked good.

6.2.12 Counter Read Messages

Counter Read Request Message



00/CTP$OPCODE — 1 Byte, value is 10/CTP$COUNTSREQ.

01/CTP$REFERENCE — 4 Bytes, contains reference number.

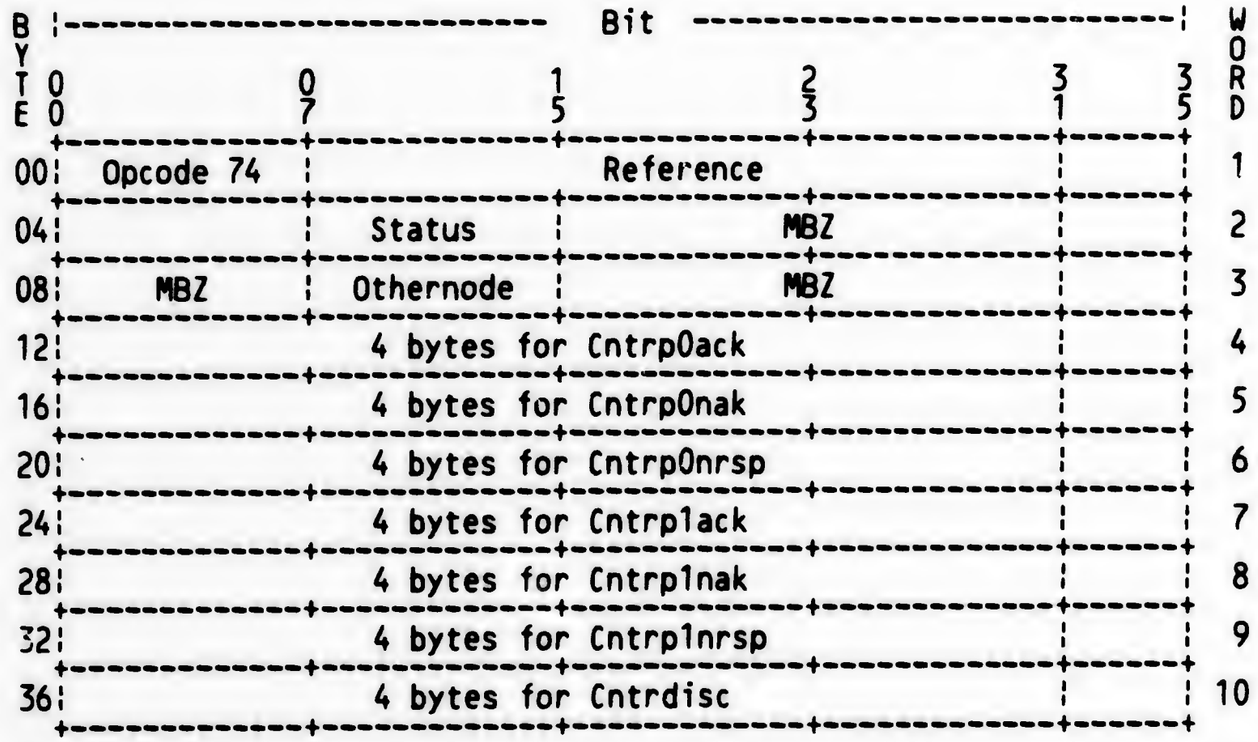
05/CTP$RESERV5 — 4 Bytes, must be zero.

09/CTP$OTHERNODE — 1 Byte containing number of node which future counts should monitor or which counts are read.

10/CTP$CNTFLG — 1 Byte indicating to keep or release control of counters. Values are:

00/CTP$RELCNT — Release counters.
01/CTP$KEEPCNT — Keep counters.

Counter Read Response Message



00/CTP$OPCODE 1 Byte, value is 74/CTP$COUNTSRSP.

01/CTP$REFERENCE 4 Bytes, contains reference number.

05/CTP$STATUS 1 Byte with the CTP status. Values are:

000.(000)/CTP$SUCCESS Function Successful.
253.(375)/CTP$NORESOURCE Resource Failure.
254.(376)/CTP$NONSENSE Request Not Understood.

06/CTP$RESERV6 3 Bytes, must be zero.

09/CTP$OTHERNODE 1 Byte containing number of node which future counts will monitor.

10/CTP$RESERV10 2 Bytes, must be zero.

12/CTP$CNTRPOACK 4 Bytes containing total ACKs received on path 0.

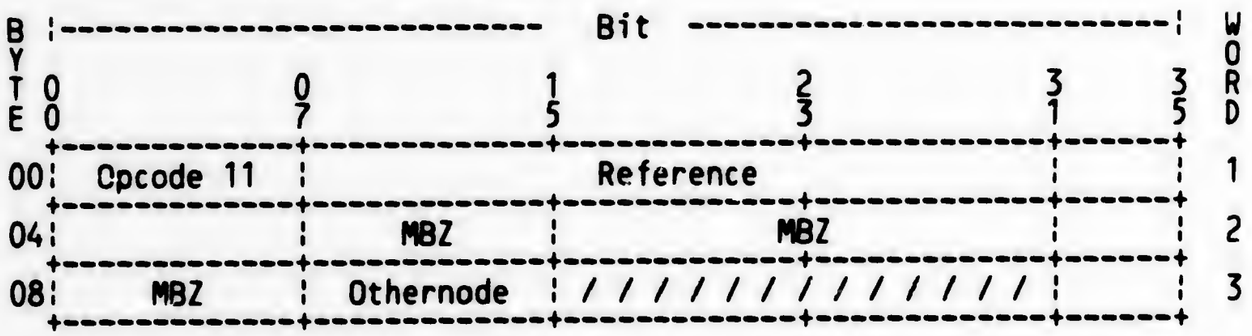
16/CTP$CNTRPONAK 4 Bytes containing total NAKs received on path 0.

20/CTP$CNTRPONRSP 4 Bytes containing total no responses

	received on path 0.
24/CTP$CNTRP1ACK	4 Bytes containing total ACKs received on path 1.
28/CTP$CNTRP1NAK	4 Bytes containing total NAKs received on path 1.
32/CTP$CNTRP1NRSP	4 Bytes containing total no responses received on path 1.
36/CTP$CNTRDISCDG	4 Bytes containing total Discarded DGs

6.2.13 Connect Messages

Connect Request Message



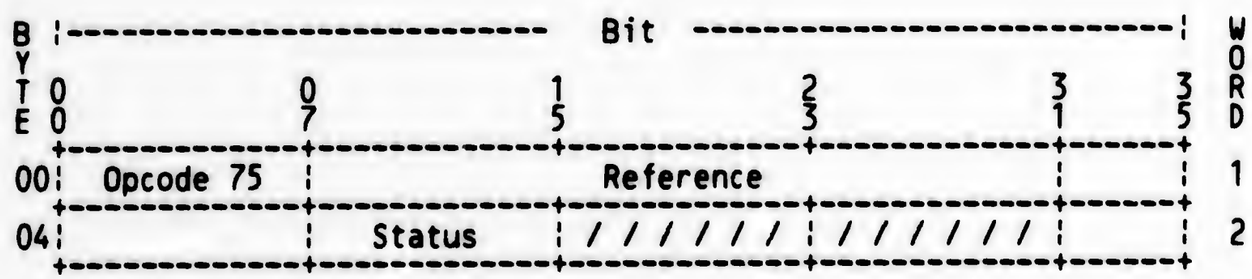
00/CTP$OPCODE 1 Byte, value is 11/CTP$CONNECTREQ.

01/CTP$REFERENCE 4 Bytes, contains reference number.

05/CTP$RESERV5 4 Bytes, must be zero.

09/CTP$OTHERNODE 1 Byte containing the node to connect to.

Connect Response Message



00/CTP$OPCODE 1 Byte, value is 75/CTP$CONNECTRSP.

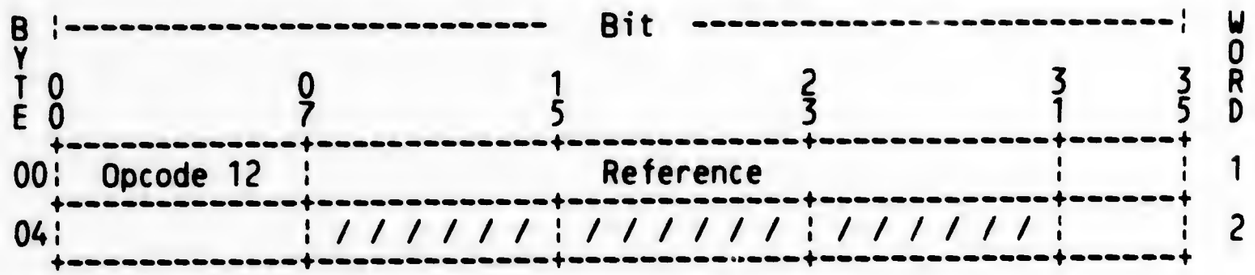
01/CTP$REFERENCE 4 Bytes, contains reference number.

05/CTP$STATUS 1 Byte, contains status of requested operation.

000.(000)/CTP$SUCCESS Function Successful.
245.(365)/CTP$CONTRANS Connection is in state of transition
248.(370)/CTP$NODEUNKNOWN Node Unknown To SCA.
251.(373)/CTP$FAILURE Function Failure.
253.(375)/CTP$NORESOURCE Resource Failure.
254.(376)/CTP$NONSENSE Request Not Understood.

6.2.14 Listener Disconnect Messages

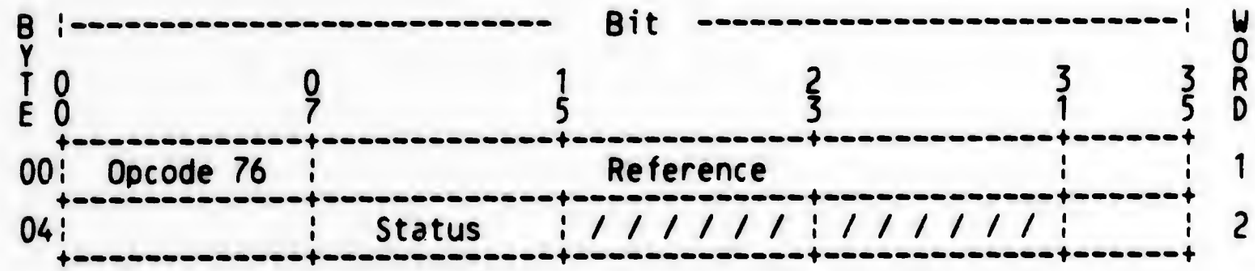
Listener Disconnect Request Message



00/CTP$OPCODE 1 Byte, value is 12/CTP$FINISHREQ.

01/CTP$REFERENCE 4 Bytes, contains reference number.

Listener Disconnect Response Message



00/CTP$OPCODE 1 Byte, value is 76/CTP$FINISHRSP.

01/CTP$REFERENCE 4 Bytes, contains reference number.

05/CTP$STATUS 1 Byte, value must be 00/CTP$SUCCESS.

```
;*****************************************************************************
;
PRODUCT CODE:           MAINDEC-10-DFCIC-A
                        VERSION  0.1
;
PRODUCT NAME:           PDP10 COMPUTER INTERCONNECT(CI) RESPONDER
;
DATE RELEASED:          OCTOBER 1984
;
UPDATE AUTHOR:          R. Stravinski
;
1.      Preliminary Release
;*****************************************************************************
;
PRODUCT CODE:           MAINDEC-10-DFCIC-B
                        VERSION  0.2
;
PRODUCT NAME:           PDP10 COMPUTER INTERCONNECT(CI) RESPONDER
;
DATE RELEASED:          November 1984
;
UPDATE AUTHOR:          R. Stravinski
;
 1. Fixed filling of data buffer in MAP buffer routine (ACT01).
;*****************************************************************************
```

037

SEQ 0050

```
;*****************************************************************************
;
; PRODUCT CODE:            MAINDEC-10-DFCIC-C
;
; VERSION:                 3
;
; PRODUCT NAME:            DFCIC - CI Cluster Test Protocol Responder
;
; DATE RELEASED:           August 21, 1985
;
; UPDATE AUTHOR:           Gregory A. Scott
;
; REASONS:                 Update for TOPS-10 operation.
;
;                          Add support for SCS. UUO and DIAG. UUO.
;                          Add code for TOPS-10 software interrupt
;                          system, getting system uptime and time of
;                          day, outputting numbers and text.
;
;                          Rework timing routine to use TIMER JSYS
;                          and PITMR UUO. The old method of using
;                          forks was not compatible with efficient
;                          operation or TOPS-10.
;
;                          Output an announcement message at startup
;                          time.
;
;*****************************************************************************
```

038

SEQ 0051

DOCUMENT
****
DFCIC Cluster Test Protocol Responder
****

COPYRIGHT 1985
DIGITAL EQUIPMENT CORPORATION
MAYNARD, MASS. 01754

TABLE OF CONTENTS
****

15 Definitions
353 Initialization Code
444 Mainline Code
542 CTP Function Set
609 CTP Buffer Map
814 CTP Buffer Unmap
928 CTP Move Buffer
1216 CTP Generate Message and Datagram
1422 CTP Generate Reset
1591 CTP Generate Start
1773 CTP Unsolicited Activity
1848 CTP Configuration
1942 CTP Counter Read
2114 CTP Connect
2203 CTP Disconnect
2233 Subroutines
3167 Send CTP Response
3220 Timeout Routines
3415 Interrupt System
3536 Return Uptime/Time Of Day
3580 Print Message Subroutines
3628 Perform SCS JSYS/UUO

TABLE OF CONTENTS
****

3767 Send Unimplemented Opcode
3817 Send No Resource
3865 Request Connection
3933 Listen for Controller
3978 Listen for Responder
4023 Accept Connection
4060 Reject Connection
4098 Close Connection
4135 Queue Message Buffer
4174 Queue Datagram Buffer
4213 Recieve Message
4255 Recieve Datagram
4297 Send Message or Datagram
4364 Connection State Poll
4426 Return Local Node Number
4459 Return Buffer Sizes
4495 Return Status
4533 Return Configuration Information
4597 Map Buffer
4633 Unmap Buffer
4670 Send Data
4711 Read Data

TABLE OF CONTENTS
****

4751 Get Data Queue Entry
4785 Get Event Queue Entry
4843 Perform DIAG JSYS or UUO
4958 TOPS-10 Interrupt Dispatch
4989 TOPS-20 Interrupt Dispatch
5042 Interrupt Routine For Inputting Datagrams
5141 Interrupt Routine For Inputting Messages
5273 Interrupt Routine For DMA Transfer Complete
5320 Interrupt Routine For Connections & Errors
5611 Interrupt System Tables
5658 Tables

15 Definitions

27

Copyright 1984, 1985
Digital Equipment Corporation
Marlboro, Massachusetts 01752

Author: Ron Stravinski
Update Author: Gregory A. Scott

353 Initialization Code

444 Mainline Code

542 CTP Function Set

609 CTP Buffer Map

814 CTP Buffer Unmap

928 CTP Move Buffer

1216 CTP Generate Message and Datagram

1422 CTP Generate Reset
1591 CTP Generate Start
1773 CTP Unsolicited Activity
1848 CTP Configuration
1942 CTP Counter Read
2114 CTP Connect
2203 CTP Disconnect
2233 Subroutines
3167 Send CTP Response
3220 Timeout Routines

```
                ;*****************************************************************************
3415            ; Interrupt System
                ;*****************************************************************************

                ;*****************************************************************************
3536            ; Return Uptime/Time Of Day
                ;*****************************************************************************

                ;*****************************************************************************
3580            ; Print Message Subroutines
                ;*****************************************************************************

                ;*****************************************************************************
3628            ; Perform SCS JSYS/UUO
                ;*****************************************************************************

                ;*****************************************************************************
3767            ; Send Unimplemented Opcode
                ;*****************************************************************************

                ;*****************************************************************************
3817            ; Send No Resource
                ;*****************************************************************************

                ;*****************************************************************************
3865            ; Request Connection
                ;*****************************************************************************

                ;*****************************************************************************
3933            ; Listen for Controller
                ;*****************************************************************************

                ;*****************************************************************************
3978            ; Listen for Responder
                ;*****************************************************************************

                ;*****************************************************************************
4023            ; Accept Connection
                ;*****************************************************************************
```

```
                ********************************************************************************
        4060    Reject Connection
                ********************************************************************************

                ********************************************************************************
        4098    Close Connection
                ********************************************************************************

                ********************************************************************************
        4135    Queue Message Buffer
                ********************************************************************************

                ********************************************************************************
        4174    Queue Datagram Buffer
                ********************************************************************************

                ********************************************************************************
        4213    Recieve Message
                ********************************************************************************

                ********************************************************************************
        4255    Recieve Datagram
                ********************************************************************************

                ********************************************************************************
        4297    Send Message or Datagram
                ********************************************************************************

                ********************************************************************************
        4364    Connection State Poll
                ********************************************************************************

                ********************************************************************************
        4426    Return Local Node Number
                ********************************************************************************

                ********************************************************************************
        4459    Return Buffer Sizes
                ********************************************************************************
```

4495 Return Status

4533 Return Configuration Information

4597 Map Buffer

4633 Unmap Buffer

4670 Send Data

4711 Read Data

4751 Get Data Queue Entry

4785 Get Event Queue Entry

4843 Perform DIAG JSYS or UUO

4958 TOPS-10 Interrupt Dispatch

```
                ;*****************************************************************
        4989    ; TOPS-20 Interrupt Dispatch
                ;*****************************************************************

                ;*****************************************************************
        5042    ; Interrupt Routine For Inputting Datagrams
                ;*****************************************************************

                ;*****************************************************************
        5141    ; Interrupt Routine For Inputting Messages
                ;*****************************************************************

                ;*****************************************************************
        5273    ; Interrupt Routine For DMA Transfer Complete
                ;*****************************************************************

                ;*****************************************************************
        5320    ; Interrupt Routine For Connections & Errors
                ;*****************************************************************

                5351    ********* Virtual circuit broken **********
                5404    ********* Connect to a listen **********
                5467    ********* Connection was accepted(for 3rd party) **********
                5479    ********* Connection was rejected(for 3rd party) **********
                5490    ********* Message or datagram send complete **********
                5551    ********* Little credit left **********
                5561    ********* Node went offline **********
                5567    ********* Node went online **********
                5573    ********* Ok to send messages **********
                5594    ********* Remote initiated disconnect **********
                5600    ********* Port broke connection **********
                5606    ********* Credit is available **********
```

```
                ;*****************************************************************************
        5611    ; Interrupt System Tables
                ;*****************************************************************************

                ;*****************************************************************************
        5658    ; Tables
                ;*****************************************************************************
```

```
                                ;Z:<GSCOTT.DFCIC>DFCIC1.MAC.84 27-Aug-85 08:17:22, Edit by GSCOTT
                                ;Add SCSBLK, SCSFCN, SCSCAL, SCSERC
                                ;Z:<GSCOTT.DFCIC>DFCIC1.MAC.79  2-Aug-85 16:06:16, Edit by GSCOTT
                                ;Turn off interrupts when processing things to do at lower level
                                ;Z:<GSCOTT.DFCIC>DFCIC1.MAC.72 31-Jul-85 16:40:54, Edit by GSCOTT
                                ;Don't smash first word of SCS UUO arg block on the 10, leave it for debugging.
                                ;Z:<GSCOTT.DFCIC>DFCIC1.MAC.69 30-Jul-85 18:54:14, Edit by GSCOTT
                                ;Output error number in diag uuo errors
                                ;Z:<GSCOTT.DFCIC>DFCIC1.MAC.65 30-Jul-85 11:38:31, Edit by GSCOTT
                                ;Fix ill memory ref on TOPS-10 cause by bad index generation at GETINA+a few.
                                ;Z:<GSCOTT.DFCIC>DFCIC1.MAC.15 16-Jul-85 13:30:13, Edit by GSCOTT
                                ;Start TOPS-10 conversion

                                        TITLE   DFCIC   Cluster Test Protocol Responder
                                        SUBTTL  Definitions

                                ;Program version

                    000014      VEDIT==14                       ;Edit number
                    000003      DECVER==3                       ;Major version

000137                                  LOC     137             ;Where versions a stored
000137  000300  000014                  <DECVER>B11+VEDIT       ;Set version number
000000'                                 RELOC                   ;Relocatable code

                                Comment ~

                                        Copyright 1984, 1985
                                        Digital Equipment Corporation
                                        Marlboro, Massachusetts 01752

                                                Author: Ron Stravinski
                                         Update Author: Gregory A. Scott

                                ~

                                ;Assembler directives

                                        SEARCH  MONSYM,UUOSYM   ;Get symbols for program use
                                        SALL                    ;Suppress macro expansions
                                        .DIREC  FLBLST          ;First line binary listing only

                                ;Define the ACs we will use today

                    000000      AC0==0
                    000001      AC1==1
                    000002      AC2==2
                    000003      AC3==3
                    000004      AC4==4
                    000005      AC5==5
                    000006      AC6==6
                    000007      AC7==7
                    000010      AC10==10
                    000011      AC11==11
                    000012      AC12==12
                    000013      AC13==13
```

0051


```
     56                 000016          AC16==16
     57
     58                 000016          CURSTS==AC16                    ;Current rhblk - status,,cblk address
     59
     60                 000017          P==17                           ;Stack ac
```

```
                        ;Macros for our use today

                        ;Macro to produce a trace message

                        DEFINE  TRMSG(TEXT),<
                                SKIPE   TRACE                   ;;Trace in effect?
                                GO      [PUT    AC1             ;;Yes, save AC1
                                        HRROI   AC1,[ASCIZ ~'TEXT
~]                                                              ;;Load the message
                                        GO      PRINTT          ;;Print on terminal
                                        GET     AC1             ;;Restore AC1
                                        RTN]                    ;;Return
                        >;end of DEFINE TRMSG

                        ;Macro to produce a fatal error message then exit

                        DEFINE  CRASH(CODE,TEXT),<
                        S..'CODE':!JRST [HRROI  AC1,[ASCIZ/? fatal error 'CODE': TEXT
                        /]
                                        GO      PRINTT          ;;Print message
                                        JRST    HALTP]          ;;Halt program
                        >;end of DEFINE CRASH

                        ;Macro to produce announce text.

                        DEFINE  PRVERS  (EDIT,DEC),<
                                ASCIZ   /DFCIC - Cluster Test Protocol Responder version 'DEC'('EDIT')
                        />;End of DEFINE PRVERS

000000' 10: 106 103 111 103     ANNOUN: PRVERS(\VEDIT,\DECVER)          ;Generate text here
```

```
                                ;Static definitons

000040                  COUNT==^D32             ;Maximum number of virtual circuits
000010                  RHBPER==^D8             ;# Of req header blks/connection block
                                                ;Ei. 4 Msg & 4 dgm buffers are queued/connection
000230                  MXDGM==^D152            ;Maximum datagram size
000004                  RHBLEN==4               ;Request header blk length
000012                  CBKLEN==^D10            ;Connection blk length
000020                  MAXOPC==^D16            ;Maximum number of opcodes
000100                  RESOFS==^D64            ;Response offset(request opcode+64)
000074                  TIMELN==^D60            ;Time(min) between spear entries
000020                  SPOFST==^D16            ;Max 15 other nodes/ci. Spear table offset.

                        ;Index symbols for the request header block(rhblk)

000000                  FLINK==0                ;Forward link(rh)
000000                  SIXCID==0               ;6 Bit cid used in sca connect(lh)
000001                  BLINK==1                ;Backward link(rh)
000001                  RPCNT==1                ;Storage for ctp repcount(lh)
000002                  STATUS==2               ;Status,,cblk address
000003                  CTPPKT==3               ;Start of ctp packet(rblk address)

                        ;Index symbols for the cblk

000000                  CID==0                  ;Connect id
000001                  CBSTS==1                ;Cblk status word
000002                  REQ1==2                 ;1St rhblk address
000003                  REQ2==3                 ;2Nd rhblk address
000004                  NODE==4                 ;Node number
000005                  LOCPRT==5               ;Local port number(ci number)
000006                  BFSCA1==6               ;32 Bit name sca gave us for
000007                  BUFF1==7                ; Our buffer(bufx) address
000010                  BFSCA2==8               ;32 Bit name sca gave us for
000011                  BUFF2==9                ; Our buffer(bufx) address
```

```
                                ;Bit definitions for rhblk(cursts) status word
                                
000400  000000                  DGMSG==1B09                     ;Datagram via Message service.
000200  000000                  CONT==1B10                      ;Continue bit(gobit) for 3rd party
000100  000000                  WAITT==1B11                     ;Waiting for a connection or a move resp
000040  000000                  OVRDUE==1B12                    ;Ctp response overdue
000020  000000                  DGM==1B13                       ;Packet was received via a datagram
000010  000000                  VCGONE==1B14                    ;Virtual vircuit has gone away
000004  000000                  ONETIM==1B15                    ;One time flag
000002  000000                  DONE==1B16                      ;Finished with this ctp packet
000001  000000                  GOBIT==1B17                     ;This ctp packet is ready to be executed

                                ;Bit definitions for cblk status word

200000  000000                  THIRD==1B1                      ;This cblk is a 3rd party connection
100000  000000                  GTINFO==1B2                     ;Store various information in the cblk
040000  000000                  DMA==1B3                        ;Dma xfer(read or write) has completed
020000  000000                  OKTOSD==1B4                     ;"OK TO SEND" Data to remote node
010000  000000                  GANRES==1B5                     ;Generate activity no resource bit
004000  000000                  GA1CON==1B6                     ;"REQ1" In cblk received its response
002000  000000                  GA2CON==1B7                     ;"REQ2" In cblk received its response

000040  000000                  OVRDUE==1B12                    ;Ctp response overdue
000010  000000                  VCGONE==1B14                    ;Virtual vircuit has gone away

                                ;Connection & listen data definitions

                000000          PCLTYP==0                       ;Protocol type
                000003          PCLVER==3                       ;Protocol version
                000000          PCLREV==0                       ;Protocol revision

                                ;Sca reject and disconnect codes

                000000          NMATCH==0                       ;No matching listen found
                000001          MATCH==1                        ;Matching listen found
                000002          NRESRC==2                       ;No resources available

                                ;Operator defined instructions

260740  000000                  OPDEF   GC      [PUSHJ  P,]     ;Subroutine call
263740  000000                  OPDEF   RTN     [POPJ   P,]     ;Subroutine return
261740  000000                  OPDEF   PUT     [PUSH   P,]     ;Put data on push list
262740  000000                  OPDEF   GET     [POP    P,]     ;Get data from pu  .  .st

                                ;Storage for the pushdown list

000014' 777577  000014' PLIST:  PLIST-PLISTE,,PLIST
000015' PLISTS: BLOCK   200
000215' 000 00 0 00 000000      PLISTE: Z                       ;End of pushdown list

000216' 000000  000000          TRACE:  0                       ;-1 = Typeout program text
```

```
                                ;Storage area for flags

000217'                         STRTF:                          ;Start of storage area for flags
000217'  000 00 0 00 000000     SPEARF: Z                       ;If 0 make a spear entry on start up
000220'  000 00 0 00 000000     INTFLG: Z                       ;Number of interrupts to service
000221'  000 00 0 00 000000     WAITF:  Z                       ;Wait jsys flag
000222'  000 00 0 00 000000     LASTF:  Z                       ;Last xfer - used in msg/dgm test
000223'  000 00 0 00 000000     RESPFG: Z                       ;-1 = Connect from a responder
000224'  000 00 0 00 000000     GENFUN: Z                       ;Storage for genfunct
000225'  000 00 0 00 000000     GENCON: Z                       ;Storage for genconst
000226'  000 00 0 00 000000     GENLEN: Z                       ;Storage for genlength
000227'  000 00 0 00 000000     BUFLEN: Z                       ;Storage for buflength
000230'  000 00 0 00 000000     BUFNAM: Z                       ;Storage for bufname
000231'  000 00 0 00 000000     OPCODE: Z                       ;Storage for opcode
000232'  000 00 0 00 000000     OTHERN: Z                       ;Storage for other node
000233'  000 00 0 00 000000     MOVETY: Z                       ;0 = Read  1 = write
000234'  000 00 0 00 000000     MONTYP: Z                       ;-1=Tops20 0=tops10
000235'  000 00 0 00 000000     CYCL60: Z                       ;Divisor for jiffies: 50 or 60

000236'                         SCSCAL: BLOCK   1               ;Remembered SCS caller
000237'                         SCSERC: BLOCK   1               ;SCS error code from last error
000240'                         SCSFCN: BLOCK   1               ;Remembered SCS function code
000241'                         SCSBLK: BLOCK   1               ;Remembered SCS arg block addr
                                                                ;SCSBLK must follow SCSFCN!
000242'  000 00 0 00 000000     DLYTIM: Z                       ;Temporary storage of the delay byte
000243'  000 00 0 00 000000     DLYRBK: Z                       ;Temporary storage of the current rhblk
000244'  000 00 0 00 000000     CONIDC: Z                       ;Connect id of a recent cntrlr listen
000245'  000 00 0 00 000000     CONIDR: Z                       ;Connect id of a recent responder listen
000246'  000 00 0 00 000000     CIDLOC: Z                       ;Used to make a cid when needed
000247'  000 00 0 00 000000     OURNOD: Z                       ;Storage for our node number
000250'  000 00 0 00 000000     WAITIM: Z                       ;3Rd party storage of the wait time
000251'  000 00 0 00 000000     WAITRB: Z                       ;3Rd party storage of the current rhblk
000252'  000 00 0 00 000000     STRTAD: Z                       ;Storage for the start address
000253'  000 00 0 00 000000     UNSACT: Z                       ;Storage for the unsolicited activity byte
000254'  000 00 0 00 000000     TIMESP: Z                       ;Time(min) of our last entry into spear
000255'  000 00 0 00 000000     SYSAC2: Z                       ;Block length for "SYERR"
000256'  000 00 0 00 000000     CIPORT: Z                       ;Our local ports can be different ci's
000257'  000 00 0 00 000000     DGMSRV: Z                       ;Count of recieved dgms
000260'  000 00 0 00 000000     MSGSRV: Z                       ;Count of recieved msgs
000261'  000 00 0 00 000000     DMAWRT: Z                       ;Count of dma writes
000262'  000 00 0 00 000000     DMARD:  Z                       ;Count of dma reads
000263'  000 00 0 00 000000     DMACMP: Z                       ;Count of dma xfers completed
000264'  000 00 0 00 000000     GAFLAG: Z                       ;Gen act test
000265'  000 00 0 00 000000     MAXMSG: Z                       ;Maximum msg xfer size
000266'  000 00 0 00 000000     MAXDGM: Z                       ;Maximum dgm xfer size
000267'  000 00 0 00 000000     SCALBF: Z                       ;Sca local buffer
000270'  000 00 0 00 000000     SCARBF: Z                       ;Sca remote buffer
000271'  000 00 0 00 000000     LBUFOF: Z                       ;Local buffer offset
000272'  000 00 0 00 000000     RBUFOF: Z                       ;Remote buffer offset
000273'  000 00 0 00 000000     DGMMSG: Z                       ;Flag for dgm under msg service
```

```
                                        ;Execution words

000274' 000 00 0 00 000000      TOPEXE: Z                       ;First ctp request to be executed(rhblk address)
000275' 000 00 0 00 000000      BOTEXE: Z                       ;Last ctp request to be executed

000276' 000 00 0 00 000000      CUREXE: Z                       ;Current address of executing rhblk

                000276'         ENDF==.-1                       ;End of storage area for flags

                                        ;Ctp status byte symbols

                000000          SUCCES==00                      ;Successful status
                000001          REALBU==01                      ;Real buffer mapped/unmapped
                000002          FAKEBU==02                      ;Black hole buffer mapped/unmapped

                                        ;Ctp status byte error symbols

                000366          NOCONN==^D246                   ;No connection exits
                000367          PKTSZF==^D247                   ;Unable to use specified packet size
                000370          NODEUN==^D248                   ;Node unknown to sca
                000371          NOBUFM==^D249                   ;No buffer mapped
                000372          BUFLIM==^D250                   ;Buffer map limit exceeded
                000373          FUNCFL==^D251                   ;Function failure
                000374          RQREFU==^D252                   ;Request refused
                000375          NORESO==^D253                   ;Resource failure
                000376          NONSEN==^D254                   ;Request not understood
                000377          NTIMPL==^D255                   ;Request not implemented


                                        ;Byte pointers for the sca jsys monitor call.

000277' 777777  000301'         CNTRLR: -1,,CNTLR
000300' 777777  000305'         RSPNDR: -1,,RSPDR


                                        ;Asciz text storage for the sca jsys monitor call.

000301' 103 124 120 044 103     CNTLR:  ASCIZ/CTP$CONTROLLER    /
000305' 103 124 120 044 122     RSPDR:  ASCIZ/CTP$RESPONDER     /
000311' 000 003 000 000 00      CONDAT: BYTE (8) PCLTYP,PCLVER,PCLREV,0 ;Two words
000312' 000 00 0 00 000000              Z                               ; Of Connection Data
```

```
                                ;Symbols for byte pointers.
                                ;When using these byte pointers ac1 must contain the rblk address.

                                ;Starting at byte 0
000313' 34 10 0 01 000000       OPCO:   POINT 8,(AC1),7                 ;1 Byte for opcode

                                ;Starting at byte 1
000314' 04 30 0 01 000000       REFE:   POINT 24,(AC1),31               ;3 Bytes for cmd reference #

                                ;Starting at byte 5
000315' 24 10 0 01 000000       BUFT:   POINT 8,(AC1),15                ;1 Byte for buffer type
000316' 24 10 0 01 000000       DELA:   POINT 8,(AC1),15                ;1 Byte for delay
000317' 24 10 0 01 000000       STAT:   POINT 8,(AC1),15                ;1 Byte for ctp status

                                ;Starting at byte 6
000320' 04 20 0 01 000000       ACTC:   POINT 16,(AC1),31               ;2 Byte for actual action count
000321' 04 20 0 01 000000       REPC:   POINT 16,(AC1),31               ;2 Byte for repeat count

                                ;Starting at byte 8
000322' 34 10 0 01 000000       EXTE:   POINT 8,(AC1),7                 ;1 Byte for defaults
000323' 34 10 0 01 000000       GENF:   POINT 8,(AC1),7                 ;1 Byte for function
000324' 34 10 0 01 000000       MOVE:   POINT 8,(AC1),7                 ;1 Byte for type of move
000325' 04 40 0 01 000000       ACTB:   POINT 32,(AC1),31               ;4 Bytes for activity buffer
000326' 04 40 0 01 000000       FCMD:   POINT 32,(AC1),31               ;4 Bytes for the failed cmd

                                ;Starting at byte 9
000327' 24 10 0 01 000000       GENC:   POINT 8,(AC1),15                ;1 Byte for fill constant
000330' 24 10 0 01 000000       PORT:   POINT 8,(AC1),15                ;1 Byte for port/sca status
000331' 24 10 0 01 000000       OTHE:   POINT 8,(AC1),15                ;1 Byte for othernode

                                ;Starting at byte 10
000332' 14 10 0 01 000000       NOAC:   POINT 8,(AC1),23                ;1 Byte for unsolicited activity action
000333' 14 10 0 01 000000       PKTZ:   POINT 8,(AC1),23                ;1 Byte for packet size

                                ;Starting at byte 11
000334' 04 10 0 01 000000       PKTM:   POINT 8,(AC1),31                ;1 Byte for packet size multiple
000335' 04 10 0 01 000000       POST:   POINT 8,(AC1),31                ;1 Byte for path 0 status

                                ;Starting at byte 12
000336' 24 20 0 01 000000       GENL:   POINT 16,(AC1),15               ;2 Byte for generated length
000337' 34 10 0 01 000000       P1ST:   POINT 8,(AC1),7                 ;1 Byte for path 1 status
000340' 04 40 0 01 000000       BUFL:   POINT 32,(AC1),31               ;4 Byte for size of buffer
000341' 04 40 0 01 000000       CNOA:   POINT 32,(AC1),31               ;4 Byte for acks on path 0
000342' 04 40 0 01 000000       STAR:   POINT 32,(AC1),31               ;4 Byte for optional start address
000343' 34 10 0 01 000000       ACTN:   POINT 8,(AC1),7                 ;1 Byte containing # of activity cmds
000344' 34 10 0 01 000000       FSTS:   POINT 8,(AC1),7                 ;1 Byte containing status of failed cmd
```

```
                                ;Starting at byte 16
000345' 04 40 0 01 000000       BUFS:   POINT 32,(AC1),31       ;4 Byte for local buffer name
000346' 04 40 0 01 000000       CNON:   POINT 32,(AC1),31       ;4 Byte for naks on path 0

                                ;Starting at byte 20
000347' 04 40 0 01 000000       BUFO:   POINT 32,(AC1),31       ;4 Byte for local buffer offset
000350' 04 40 0 01 000000       CONR:   POINT 32,(AC1),31       ;4 Byte for path 0 no response

                                ;Starting at byte 24
000351' 04 40 0 01 000000       BUFD:   POINT 32,(AC1),31       ;4 Byte for remote buffer name
000352' 04 40 0 01 000000       CN1A:   POINT 32,(AC1),31       ;4 Byte for acks on path 1

                                ;Starting at byte 28
000353' 04 40 0 01 000000       BUDO:   POINT 32,(AC1),31       ;4 Byte for remote buffer offset
000354' 04 40 0 01 000000       CN1N:   POINT 32,(AC1),31       ;4 Byte for naks on path 1

                                ;Starting at byte 32
000355' 04 40 0 01 000000       C1NR:   POINT 32,(AC1),31       ;4 Byte for path 1 no response

                                ;Starting at byte 36
000356' 04 40 0 01 000000       DSDG:   POINT 32,(AC1),31       ;4 Byte for total discarded dgm's



                                ;Symbols for "SPEAR" table byte pointers.
                                ;When using these byte pointers ac1 must contain a spear table address.

000357' 00 11 0 01 000000       SPRBLN: POINT   9,(AC1),35      ;Block length for responder stat block

000360' 11 03 0 01 000000       SPRCIN: POINT   3,(AC1),26      ;Ci number

000361' 00 04 0 01 000000       SPRNOD: POINT   4,(AC1),35      ;Node number
```

```
                                        SUBTTL  Initialization Code

                                ;The following code is executed only once (during start up).

000362' 200 17 0 00 000014'     START:  MOVE    P,PLIST                 ;Initialize pushdown list

                                ;Init the flags area

000363' 402 00 0 00 000217'             SETZM   STRTF                   ;Clear 1st flag
000364' 200 00 0 00 261040              MOVE    AC0,[STRTF,,STRTF+1]    ;Start of blt
000365' 251 00 0 00 000276'             BLT     AC0,ENDF                ;End location of blt

                                ;Figure out the monitor type first

000366' 200 01 0 00 261041              MOVE    AC1,[%CNMNT]            ;Load %CNMNT monitor type word
000367' 047 01 0 00 000041              GETTAB  AC1,                    ;This will not call PA1050 on the -20
000370' 400 01 0 00 000000               SETZ   AC1,                    ;Oops
000371' 602 01 0 00 040000              TRNE    AC1,4B23                ;Skip if TOPS-10
000372' 334 01 0 00 261042              SKIPA   AC1,[RESET%]            ;TOPS-20, load a twenty reset
000373' 334 01 0 00 261043              SKIPA   AC1,[CALLI 0]           ;TOPS-10, load a ten reset
000374' 476 00 0 00 000234'             SETOM   MONTYP                  ;TOPS-20, set the flag indicating that
000375' 256 00 0 00 000001              XCT     AC1                     ;Reset the world

                                ;Announce ourselves.

000376' 561 01 0 00 000000'             HRROI   AC1,ANNOUN              ;Point to announce text
000377' 260 17 0 00 004550'             GO      PRINTT                  ;Print that

                                ;Initialize the execution header words.

000400' 201 00 0 00 000274'             MOVEI   AC0,TOPEXE              ;Get header address and
000401' 202 00 0 00 000274'             MOVEM   AC0,TOPEXE              ; Store it in the top & bottom saying
000402' 202 00 0 00 000275'             MOVEM   AC0,BOTEXE              ; No request blocks have been received

                                ;Enable capabilities

000403' 260 17 0 00 004472'             GO      ENABLE                  ;Enable your priviledges

                                ;Set up an interrupt channels and system.  TOPS-20: Enable interrupt channel
                                ;for TIMER% and 4 channels for SCS% interrupts (datagram available, message
                                ;available, DMA transfer complete, virtual connections and SCA related errors).
                                ;TOPS-10: two interrupt vectors are used, one for SCS. UUO and one for PITMR.
                                ;UUO interrupts.

000404' 260 17 0 00 004360'             GO      IFPINT                  ;Inferior process interrupt
```

```
                                ;Here we "LISTEN" for a "CONNECT" from a controller and a responder for the
                                ;first time.  When this "LISTEN" is answered with a "CONNECT", another "LISTEN"
                                ;is executed but from the interrupt routine.

000405' 260 17 0 00 005136'             GO      LISTNC                  ;Listen for an connect from controller
000406' 260 17 0 00 005177'             GO      LISTNR                  ;Listen for an connect from responder

                                ;Get our node number and store it in ournod.

000407' 260 17 0 00 005635'             GO      LOCALN                  ;Get local node number
000410' 201 01 0 00 005657'             MOVEI   AC1,LOCARG              ;Use local node arg block for index
000411' 200 00 0 01 000001              MOVE    AC0,.SQLNN(AC1)         ;Get our node number
000412' 202 00 0 00 000247'             MOVEM   AC0,OURNOD              ; And store it

                                ;Also store our node number in the spear block

000413' 137 00 0 00 261044              DPB     AC0,[POINT 4,SYSE10+4,35]

                                ;Get the maximum buffer transfer sizes for "MSG" and "DGM".

000414' 260 17 0 00 005661'             GO      BUFFSZ                  ;Do buffer size jsys
000415' 201 01 0 00 005704'             MOVEI   AC1,BUFARG              ;Use buffer size arg block for index
000416' 200 00 0 01 000001              MOVE    AC0,.SQLMG(AC1)         ;Get max "MSG" buffer size in words
000417' 242 00 0 00 000002              LSH     AC0,2                   ;Now convert words to bytes
000420' 202 00 0 00 000265'             MOVEM   AC0,MAXMSG              ; And store it
000421' 200 00 0 01 000002              MOVE    AC0,.SQLDG(AC1)         ;Get max "DGM" buffer size in words
000422' 242 00 0 00 000002              LSH     AC0,2                   ;Now convert words to bytes
000423' 202 00 0 00 000266'             MOVEM   AC0,MAXDGM              ; And store it

                                ;Get our port number and store it in the spear block.

000424' 402 00 0 00 000001              SETZM   AC1                     ;Zero cid
000425' 201 02 0 00 000001              MOVEI   AC2,1                   ;Use node # of 1
000426' 260 17 0 00 005737'             GO      CONFIG                  ;Do the config jsys
000427' 255 00 0 00 000000              JFCL                            ;Ignore error(unlikely)
000430' 201 01 0 00 005767'             MOVEI   AC1,CNFARG              ;Use config argument block for index
000431' 200 00 0 01 000023              MOVE    AC0,.SQLPN(AC1)         ;Get the port number
000432' 201 01 0 00 007553'             MOVEI   AC1,SYSE10+4            ;Use spear block for index
000433' 137 00 0 00 000360'             DPB     AC0,SPRCIN              ;Port# to spear table

                                ; Set up a time in location "TIMESP" which will be used for spear entries.

000434' 260 17 0 00 004507'             GO      UTIME                   ;Get the uptime in milliseconds
000435' 231 01 0 00 165140              IDIVI   AC1,^D1000*^D60         ;Change ms to minutes
000436' 202 01 0 00 000254'             MOVEM   AC1,TIMESP              ;Store the time count in minutes
```

```
                                SUBTTL  Mainline Code

                        ;This mainline routine runs in a loop looking for ctp request packets that are
                        ;awaiting execution.  If their aren't any request packets awaiting execution we
                        ;see if its time to make another spear entry, and see if any connections are
                        ;opened.  If no connections are opened we disconnect all our connections to any
                        ;3rd party responder.  We then suspend execution of this program until we get
                        ;interrupted.

000437' 402 00 0 00 000221'     MAINLN: SETZM   WAITF           ;No longer waiting for interrupt
000440' 260 17 0 00 004425'             GO      DISINT          ;Diable interrupt system
000441' 200 17 0 00 000014'             MOVE    P,PLIST         ;Safe to reset stack now
000442' 200 00 0 00 000274'             MOVE    AC0,TOPEXE      ;Get top of execution list
000443' 306 00 0 00 000274'             CAIN    AC0,TOPEXE      ;Have we received any requests yet?
000444' 254 00 0 00 000463'             JRST    MAINC           ;No
000445' 202 00 0 00 000276'             MOVEM   AC0,CUREXE      ;Yes, store currently executing routine
000446' 260 17 0 00 003706'             GO      GETINF          ;Store info in cblk

000447' 200 01 0 00 000276'     MAINA:  MOVE    AC1,CUREXE      ;Get rhblk address
000450' 200 00 0 01 000002              MOVE    AC0,STATUS(AC1) ;Get rhblk status word
000451' 603 00 0 00 000001              TLNE    AC0,(GOBIT)     ;Is go set?
000452' 260 17 0 00 000502'             GO      EXECT           ;Yes, request can be executed
000453' 260 17 0 00 003164'             GO      RMCBLK          ;Check if any cblks should be removed
000454' 200 01 0 00 000276'             MOVE    AC1,CUREXE      ;Get rhblk address
000455' 200 00 0 01 000000              MOVE    AC0,FLINK(AC1)  ;Get flink pointer to the next rhblk
000456' 323 00 0 00 000463'             JUMPLE  AC0,MAINC       ;If -1 buffer has only been queued
000457' 306 00 0 00 000274'             CAIN    AC0,TOPEXE      ;Have any more requests to do?
000460' 254 00 0 00 000463'             JRST    MAINC           ;No
000461' 202 00 0 00 000276'             MOVEM   AC0,CUREXE      ;Store currently executing routine
000462' 254 00 0 00 000447'             JRST    MAINA           ;Now look for a rhblk with "GO" set

000463' 260 17 0 00 003164'     MAINC:  GO      RMCBLK          ;Check if any cblks should be removed
000464' 260 17 0 00 003263'             GO      KIL3RD          ;Kill 3rd party conn, Deq all rhblks
000465' 260 17 0 00 004005'             GO      SPEAR           ;See if we should make a spear entry
000466' 336 00 0 00 000220'             SKIPN   INTFLG          ;Have any interrupts occured ?
000467' 254 00 0 00 000472'             JRST    MAINC1          ; No
000470' 370 00 0 00 000220'             SOS     INTFLG          ;Decrement interrupt count
000471' 254 00 0 00 000437'             JRST    MAINLN          ;Off we go

                        ;The WAIT jsys (or HIBER forever UUO) dismisses this program from getting any
                        ;runtime indefinitely.  The only way we get out of the wait state is by getting
                        ;interrupted.  When the interrupt is finished we come back to the wait state,
                        ;unless we change the return PC word in the interrupt routine. We do this in
                        ;the "VCBRKN", "INPDGM" "INPMSG", "MDSNDC", "INFPT" And "DMAXFR" interrupt
                        ;routines.  We change the pc to "MAINLN".

000472' 476 00 0 00 000221'     MAINC1: SETOM   WAITF           ;Set the wait flag
000473' 260 17 0 00 004440'             GO      ENAINT          ;Enable interrupts
000474' 332 00 0 00 000234'             SKIPE   MONTYP          ;Skip if TOPS-10
000475' 104 00 0 00 000306              WAIT%                   ;Stay here til we get an interrupt
000476' 201 01 0 00 000000              MOVEI   AC1,0           ;Load infinate wait time
000477' 047 01 0 00 000072              HIBER   AC1,            ;ZZZ...
000500' 255 00 0 00 000000               JFCL                   ;Should never return
000501' 254 00 0 00 000437'             JRST    MAINLN          ; but if it does, loop
```

```
                                ;This is where we actually dispatch to a subroutine to execute a particular ctp
                                ;request command.  First we load the ac register "CURSTS" with the status(lh)
                                ;and the cblk address(rh) of this ctp request.  Then we check to see if this
                                ;opcode is valid.  If not, we return a 17 in ac2 which will execute the
                                ;"NOTIMP" routine.  If the opcode is a valid one we use the opcode in ac2 as an
                                ;index pointer.

000502'                         EXECT:
                                ;(12)   GO      DISINT                  ;Disable the interrupt system
000502' 261 17 0 00 000001              PUT     AC1                     ;Save ac's
000503' 261 17 0 00 000002              PUT     AC2
000504' 200 01 0 00 000276'             MOVE    AC1,CUREXE              ;Get rhblk address
000505' 200 16 0 01 000002              MOVE    CURSTS,STATUS(AC1)      ;Get rhblk status word in the ac
000506' 260 17 0 00 003521'             GO      OPCHK                   ;Check if opcode is a valid one
000507' 260 17 1 02 000515'             GO      @TSTTBL(AC2)            ;Do the ctp function
000510' 200 01 0 00 000276'             MOVE    AC1,CUREXE              ;Get rhblk address
000511' 202 16 0 01 000002              MOVEM   CURSTS,STATUS(AC1)      ;Restore rhblk status word
000512' 262 17 0 00 000002              GET     AC2                     ;Restore ac's
000513' 262 17 0 00 000001              GET     AC1
                                ;(12)   GO      ENAINT                  ;Enable the interrupt system
000514' 263 17 0 00 000000              RTN                             ;Return


                                ;Dispatch table for ctp requests
                                                                        ;Opcode - function
000515' 000000  000537'         TSTTBL: FUNCST                          ;00 - Function rsp - 38  mandatory
000516' 000000  000607'                 MAPBUF                          ;01 - Buff map rsp - 20
000517' 000000  001034'                 UNMAPB                          ;02 - Buf unmp rsp - 20
000520' 000000  001165'                 MOVEBF                          ;03 - Move buf rsp - 32
000521' 000000  001477'                 GENMSG                          ;04 - Gen msg  rsp - 128 mandatory
000522' 000000  001477'                 GENDGM                          ;05 - Gen dtgm rsp - 512 mandatory
000523' 000000  001717'                 GENRST                          ;06 - Gen rest rsp - 11
000524' 000000  002114'                 GENSTR                          ;07 - Gen strt rsp - 16
000525' 000000  002320'                 SSUNSO                          ;08 - S/s unso rsp - 06  mandatory
000526' 000000  002410'                 CNFDAT                          ;09 - Con data rsp - 13  mandatory
000527' 000000  002515'                 CNTRRD                          ;10 - Cntr rd  rsp - 40  mandatory
000530' 000000  002714'                 CTPCON                          ;11 - Connect  rsp - 06
000531' 000000  003027'                 LISDIS                          ;12 - Lstr dis rsp - 06  mandatory
000532' 000000  003337'                 NOTIMP                          ;13 - Mbuf map rsp - 20
000533' 000000  003337'                 NOTIMP                          ;14 - Mbf umap rsp - 20
000534' 000000  003337'                 NOTIMP                          ;15 - Mov mbuf rsp - 28
000535' 000000  003337'                 NOTIMP                          ;16 - Mn state rsp - 06
000536' 000000  003337'                 NOTIMP                          ;Non valid ctp opcode
```

```
                                        SUBTTL  CTP Function Set

                                ;This routine tells the controller what set of opcodes this responder
                                ;Has implemented. We send back the table "OPCARY" as part of the
                                ;Response message, each bit represents an implemented opcode.

000537' 332 00 0 00 000216'     FUNCST: TRMSG   (Opcode: Function set)
000540' 260 17 0 00 261052
000541' 260 17 0 00 003543'             GO      CHGOPC                  ;Change opcode from a req to a res.
000542' 260 17 0 00 000553'             GO      VALOO                   ;Check validity of the ctp request
000543' 260 17 0 00 000560'             GO      ACTOO                   ;Perform action for the request
000544' 661 16 0 00 000002              TLO     CURSTS,(DONE)           ;Set the done bit
000545' 621 16 0 00 000001              TLZ     CURSTS,(GOBIT)          ;Clear the gobit
000546' 200 01 0 00 000276'             MOVE    AC1,CUREXE              ;Get the rhblk
000547' 202 16 0 01 000002              MOVEM   CURSTS,STATUS(AC1)      ;Store the status
000550' 260 17 0 00 004124'             GO      SEND                    ;Do the scs% jsys
000551' 255 00 0 00 000000              JFCL                            ;Error return
000552' 263 17 0 00 000000              RTN                             ;Return

                                ;Load the ctp response status field.

000553' 261 17 0 00 000000      VALOO:  PUT     AC0                     ;Save ac's
000554' 201 00 0 00 000000              MOVEI   AC0,SUCCES              ;Zero the
000555' 260 17 0 00 003450'             GO      LDSTUS                  ; Status byte in the ctp packet
000556' 262 17 0 00 000000              GET     AC0                     ;Restore ac's
000557' 263 17 0 00 000000              RTN                             ;Return

                                ;Put the "OPCARY" table in the ctp response message.

000560' 261 17 0 00 000000      ACTOO:  PUT     AC0                     ;Save ac's
000561' 261 17 0 00 000001              PUT     AC1
000562' 200 01 0 00 000276'             MOVE    AC1,CUREXE              ;Get the rhblk
000563' 200 01 0 01 000003              MOVE    AC1,CTPPKT(AC1)         ;Get the rblk address
000564' 350 00 0 00 000001              AOS     AC1                     ;Make it rblk+1
000565' 200 00 0 00 000577'             MOVE    AC0,OPCARY              ;Get start of 256 bit array
000566' 436 00 1 00 000001              IORM    AC0,@AC1                ;Set in bytes 6 & 7
000567' 505 00 0 00 000600'             HRLI    AC0,OPCARY+1            ;Start with byte 8 or 3rd word
000570' 350 00 0 00 000001              AOS     AC1                     ;Increment start address for blt
000571' 540 00 0 00 000001              HRR     AC0,AC1                 ;Copy to the buffer area
000572' 271 01 0 00 000007              ADDI    AC1,^D7                 ;Xfer 7 words
000573' 251 00 1 00 000001              BLT     AC0,@AC1                ;Do the block xfer
000574' 262 17 0 00 000001              GET     AC1                     ;Restore ac
000575' 262 17 0 00 000000              GET     AC0
000576' 263 17 0 00 000000              RTN                             ;Return
```

```
                                ;This table is a 256 bit array used in the "RESPONDER FUNCTION SET" response.
                                ;The start of this table is actually the 6th byte of the response packet.
                                ;The 1st word represents the ctp request commands used by this responder.
                                ;The 3rd word represents the ctp response commands used by this responder.

                                ;1St word = !reference #(bits 0-7)!status(bits 8-15)!256 bit array! starts at bit 16!

000577' 000003  770760          OPCARY: 3,,770760                       ;Each bit represents a
000600' 000000  000000                  0                               ; Request opcode
000601' 000003  770760                  3,,770760                       ;Each bit represents a
000602' 000000  000000                  0                               ; Response opcode
000603' 000 00 0 00 000000              Z
000604' 000 00 0 00 000000              Z
000605' 000 00 0 00 000000              Z
000606' 000 00 0 00 000000              Z

                                ;Bit definition of the 1st and 3rd words of opcary:
                                ;
                                ;Mask ---       1 1 ,, 1 1 1 ! 1 1 1 ! 0 0 0 ! 1 1 1 ! 1 1 0 ! 0 0 0
                                ;
                                ;Opcode ---     0 0 ,, 0 0 0 ! 0 0 0 ! n n n ! 1 1 1 ! 0 0 - ! - - -
                                ;       ---     7 6 ,, 5 4 3 ! 2 1 0 ! U u u ! 2 1 0 ! 9 8 - ! - - -
```

```
                                        SUBTTL  CTP Buffer Map

                                ;In this routine we map a buffer for the controlier to use.
                                ;We get an unused buffer(bufx) from our resource and give it to sca.
                                ;Sca gives us a name associated with that buffer. We in turn give
                                ;That name to the controller.

000607' 332 00 0 00 000216'     MAPBUF: TRMSG   (Opcode: Buffer map)
000610' 260 17 0 00 261064
000611' 260 17 0 00 003543'             GO      CHGOPC                  ;Change opcode from a req to a res.
000612' 260 17 0 00 000625'             GO      VAL01                   ;Check the validity of the ctp request
000613' 254 00 0 00 000622'             JRST    MAPBU1                  ;Error in the validity check
000614' 260 17 0 00 000701'             GO      ACT01                   ;Perform action for the request
000615' 255 00 0 00 000000              JFCL                            ;Error return
000616' 661 16 0 00 000002              TLO     CURSTS,(DONE)           ;Set the done bit
000617' 621 16 0 00 000001              TLZ     CURSTS,(GOBIT)          ;Clear the gobit
000620' 200 01 0 00 000276'             MOVE    AC1,CUREXE              ;Get the rhblk
000621' 202 16 0 01 000002              MOVEM   CURSTS,STATUS(AC1)      ;Store the status
000622' 260 17 0 00 004124'     MAPBU1: GO      SEND                    ;Send the ctp response message
000623' 255 00 0 00 000000              JFCL                            ;Virtual circuit gone return
000624' 263 17 0 00 000000              RTN                             ;Return
```

```
                                        ;Check the validity of the ctp request
                                
000625' 261 17 0 00 000000      VAL01:  PUT     AC0                     ;Save ac's
000626' 261 17 0 00 000001              PUT     AC1
000627' 261 17 0 00 000002              PUT     AC2
000630' 261 17 0 00 000003              PUT     AC3
000631' 200 01 0 00 000276'             MOVE    AC1,CUREXE              ;Get rhblk
000632' 200 03 0 01 000003              MOVE    AC3,CTPPKT(AC1)         ;Now get the rblk
000633' 201 01 0 03 000001              MOVEI   AC1,1(AC3)              ;Get buftype address word
000634' 135 00 0 00 000315'             LDB     AC0,BUFT                ;Get buftype
000635' 302 00 0 00 000000              CAIE    AC0,0                   ;Value 0?
000636' 254 00 0 00 000641'             JRST    VAL01G                  ;No
000637' 201 00 0 00 000376              MOVEI   AC0,NONSEN              ;Yes, do request not understood
000640' 254 00 0 00 000671'             JRST    VAL01X                  ;Take the +1 error return
000641' 307 00 0 00 000002      VAL01G: CAIG    AC0,2                   ;Value bigger than 2?
000642' 254 00 0 00 000645'             JRST    VAL01A                  ;No, continue
000643' 201 00 0 00 000376              MOVEI   AC0,NONSEN              ;Yes, do request not understood
000644' 254 00 0 00 000671'             JRST    VAL01X                  ;Take the +1 error return
000645' 260 17 0 00 003450'     VAL01A: GO      LDSTUS                  ;Go load status value of 1 or 2
000646' 201 01 0 03 000002              MOVEI   AC1,2(AC3)              ;Get genfunct address word
000647' 135 00 0 00 000323'             LDB     AC0,GENF                ;Get genfunct
000650' 307 00 0 00 000002              CAIG    AC0,2                   ;Value bigger than 2?
000651' 254 00 0 00 000654'             JRST    VAL01C                  ;No, continue
000652' 201 00 0 00 000376              MOVEI   AC0,NONSEN              ;Yes, do request not understood
000653' 254 00 0 00 000671'             JRST    VAL01X                  ;Take the +1 error return

                                        ;Check to see is if we already have 2 buffers mapped.

000654' 550 01 0 00 000016      VAL01C: HRRZ    AC1,CURSTS              ;Get cblk
000655' 332 00 0 01 000006              SKIPE   BFSCA1(AC1)             ;Is this sca buffer being used?
000656' 330 00 0 01 000010              SKIPN   BFSCA2(AC1)             ;Yes, is this sca buffer being used?
000657' 254 00 0 00 000662'             JRST    VAL01D                  ;No, we have room to map another
000660' 201 00 0 00 000372              MOVEI   AC0,BUFLIM              ;Yes, do buffer map limit exceeded
000661' 254 00 0 00 000671'             JRST    VAL01X                  ;Take the +1 error return

                                        ;Check to see if "BUFLENGTH" is to big.

000662' 201 01 0 03 000003      VAL01D: MOVEI   AC1,3(AC3)              ;Get buflength address word
000663' 135 00 0 00 000340'             LDB     AC0,BUFL                ;Get buflength
000664' 260 17 0 00 003463'             GO      P11T10                  ;Change pdp11 word to pdp10 word
000665' 202 00 0 00 000227'             MOVEM   AC0,BUFLEN              ;Store the buffer length
000666' 307 00 0 00 002000              CAIG    AC0,^D1024              ;Over max no of words?
000667' 254 00 0 00 000673'             JRST    VAL01S                  ;No, take the +2 return
000670' 201 00 0 00 000375              MOVEI   AC0,NORESO              ;Yes, do resource failure

000671' 260 17 0 00 003450'     VAL01X: GO      LDSTUS                  ;Load the status byte in the rblk
000672' 334 00 0 00 000000              SKIPA                           ;Take the +1 error return
000673' 350 00 0 17 777774      VAL01S: AOS     -4(P)                   ;+2 Return
000674' 262 17 0 00 000003              GET     AC3
000675' 262 17 0 00 000002              GET     AC2
000676' 262 17 0 00 000001              GET     AC1
000677' 262 17 0 00 000000              GET     AC0
000700' 263 17 0 00 000000              RTN                             ;Return
```

```
                                        ;Do the function the ctp request packet demanded
                                        
000701' 261 17 0 00 000000      ACT01:  PUT     AC0                     ;Save ac's
000702' 261 17 0 00 000001              PUT     AC1
000703' 261 17 0 00 000002              PUT     AC2
000704' 261 17 0 00 000003              PUT     AC3
000705' 261 17 0 00 000004              PUT     AC4
000706' 261 17 0 00 000005              PUT     AC5
                                        
                                        ;Get an available buffer(bufx).
                                        
000707' 260 17 0 00 003243'             GO      BUFAV                   ;Get a buffer
000710' 202 01 0 00 000230'             MOVEM   AC1,BUFNAM              ;Store buffer name
000711' 332 00 0 00 000001              SKIPE   AC1                     ;Any buffers available?
000712' 254 00 0 00 000715'             JRST    ACT01A                  ;Yes, continue
000713' 201 00 0 00 000375              MOVEI   AC0,NORESO              ;No, do resource failure
000714' 254 00 0 00 001022'             JRST    ACT01X                  ;Exit, take +1 error rtn
                                        
                                        ;Get a buffer name from sca.
                                        
000715' 200 01 0 00 000001      ACT01A: MOVE    AC1,AC1                 ;Memory address(bufx)
000716' 200 02 0 00 000227'             MOVE    AC2,BUFLEN              ;Size of buffer
000717' 260 17 0 00 006013'             GO      GETBUF                  ;Get the sca buffer name
000720' 254 00 0 00 000740'             JRST    ACT01M                  ;Error - do noresources
000721' 202 01 0 00 000005              MOVEM   AC1,AC5                 ;Store buffer name in ac5
                                        
                                        ;Make this buffer(bufx) unavailable to everyone else.
                                        
000722' 476 00 1 00 000230'             SETOM   @BUFNAM                 ;-1 to Bufx
                                        
                                        ;Now store both buffer names in the cblk.
                                        
000723' 550 01 0 00 000016              HRRZ    AC1,CURSTS              ;Get the cblk
000724' 332 00 0 01 000006              SKIPE   BFSCA1(AC1)             ;1St location empty?
000725' 254 00 0 00 000732'             JRST    ACT01L                  ;No, try 2nd location
000726' 202 05 0 01 000006              MOVEM   AC5,BFSCA1(AC1)         ;Yes, store the sca buffer name
000727' 200 04 0 00 000230'             MOVE    AC4,BUFNAM              ;Get our buffer name(bufx)
000730' 202 04 0 01 000007              MOVEM   AC4,BUFF1(AC1)          ; And store it
000731' 254 00 0 00 000742'             JRST    ACT01N                  ;Continue
000732' 332 00 0 01 000010      ACT01L: SKIPE   BFSCA2(AC1)             ;2Nd location empty?
000733' 254 00 0 00 000740'             JRST    ACT01M                  ;No, do no resources
000734' 202 05 0 01 000010              MOVEM   AC5,BFSCA2(AC1)         ;Yes, store the sca buffer name
000735' 200 04 0 00 000230'             MOVE    AC4,BUFNAM              ;Get our buffer name(bufx)
000736' 202 04 0 01 000011              MOVEM   AC4,BUFF2(AC1)          ; And store it
000737' 254 00 0 00 000742'             JRST    ACT01N                  ;Continue
000740' 201 00 0 00 000375      ACT01M: MOVEI   AC0,NORESO              ;No, do resource failure
000741' 254 00 0 00 001022'             JRST    ACT01X                  ;Exit, take +1 error rtn
```

```
                                        ;Now store the buffer name in the ctp packet.

000742' 200 00 0 00 000005      ACT01N: MOVE    AC0,AC5                 ;Get buffer name in ac0
000743' 242 00 0 00 777774              LSH     AC0,-4                  ;Right justify it
000744' 260 17 0 00 003502'             GO      P10T11                  ;Change name to pdp11 format
000745' 200 01 0 00 000276'             MOVE    AC1,CUREXE              ;Get rhblk
000746' 200 03 0 01 000003              MOVE    AC3,CTPPKT(AC1)         ;Now get the rblk
000747' 201 01 0 03 000004              MOVEI   AC1,4(AC3)              ;Get buflname address word
000750' 137 00 0 00 000345'             DPB     AC0,BUFS                ;Store buffer name in the ctp packet

                                        ;Get genfunct and store it

000751' 200 01 0 00 000276'             MOVE    AC1,CUREXE              ;Get rhblk
000752' 200 03 0 01 000003              MOVE    AC3,CTPPKT(AC1)         ;Now get the rblk
000753' 201 01 0 03 000002              MOVEI   AC1,2(AC3)              ;Get genfunct address word
000754' 135 00 0 00 000323'             LDB     AC0,GENF                ;Get genfunct
000755' 202 00 0 00 000224'             MOVEM   AC0,GENFUN              ;Store it

                                        ;Get genconst and store it

000756' 135 00 0 00 000327'             LDB     AC0,GENC                ;Get genconst
000757' 202 00 0 00 000225'             MOVEM   AC0,GENCON              ;Store it

                                        ;Get buflength and store it

000760' 201 01 0 03 000003              MOVEI   AC1,3(AC3)              ;Get buflength address word
000761' 135 00 0 00 000340'             LDB     AC0,BUFL                ;Get buflength
000762' 260 17 0 00 003463'             GO      P11T10                  ;Change pdp11 word to pdp10 word
000763' 202 00 0 00 000227'             MOVEM   AC0,BUFLEN              ;Store it

                                        ;Here we fill the buffer with one of the following: a constant,
                                        ;Byte pairs, or the responder node number.

000764' 200 04 0 00 000230'             MOVE    AC4,BUFNAM              ;Get bufx
000765' 201 00 0 00 000000              MOVEI   AC0,0                   ;Get a value
000766' 312 00 0 00 000224'             CAME    AC0,GENFUN              ;Fill buffer with constant?
000767' 254 00 0 00 000775'             JRST    ACT01B                  ;No
```

```
                                        ;Fill buffer with constant

000770' 200 03 0 00 000227'             MOVE    AC3,BUFLEN              ;Yes, get size of bufsiz in bytes
000771' 200 00 0 00 000225'             MOVE    AC0,GENCON              ;Get the fill constant
000772' 200 02 0 00 261071              MOVE    AC2,[POINT 8,(AC4)]     ;Init pointer
000773' 136 00 0 00 000002              IDPB    AC0,AC2                 ;Deposit the fill constant in the byte
000774' 367 03 0 00 000773'             SOJG    AC3,.-1                 ;Fill the buffer
000775' 201 00 0 00 000001      ACT01B: MOVEI   AC0,1                   ;Get a value
000776' 312 00 0 00 000224'             CAME    AC0,GENFUN              ;Fill buffer with byte pair numbers?
000777' 254 00 0 00 001011'             JRST    ACT01C                  ;No

                                        ;Fill buffer with byte pairs

001000' 200 03 0 00 000227'             MOVE    AC3,BUFLEN              ;Yes, get size of bufsiz in bytes
001001' 201 05 0 00 000000              MOVEI   AC5,0                   ;Get the 1st byte pair number
001002' 200 02 0 00 261071              MOVE    AC2,[POINT 8,(AC4)]     ;Init pointer
001003' 200 00 0 00 000005      ACT01H: MOVE    AC0,AC5                 ;Get 1st byte pair
001004' 136 00 0 00 000002              IDPB    AC0,AC2                 ;Deposit the 1st byte
001005' 242 00 0 00 777770              LSH     AC0,-^D8                ;Go to 2nd byte
001006' 136 00 0 00 000002              IDPB    AC0,AC2                 ;Deposit the 2nd byte
001007' 350 00 0 00 000005              AOS     AC5                     ;Increment byte pair number
001010' 367 03 0 00 001003'             SOJG    AC3,ACT01H              ;Fill up the buffer
001011' 201 00 0 00 000002      ACT01C: MOVEI   AC0,2                   ;Get a value
001012' 312 00 0 00 000224'             CAME    AC0,GENFUN              ;Fill buffer with responder node number?
001013' 254 00 0 00 001024'             JRST    ACT01S                  ;No

                                        ;Fill buffer with the responder node number

001014' 200 03 0 00 000227'             MOVE    AC3,BUFLEN              ;Yes, get size of bufsiz in bytes
001015' 200 00 0 00 000247'             MOVE    AC0,OURNOD              ;Get our node number
001016' 200 02 0 00 261071              MOVE    AC2,[POINT 8,(AC4)]     ;Init pointer
001017' 136 00 0 00 000002              IDPB    AC0,AC2                 ;Deposit the node number in the byte
001020' 367 03 0 00 001017'             SOJG    AC3,.-1                 ;Fill up the buffer
001021' 254 00 0 00 001024'             JRST    ACT01S                  ;No

001022' 260 17 0 00 003450'     ACT01X: GO      LDSTUS                  ;Load the status byte
001023' 334 00 0 00 000000              SKIPA                           ;Take the +1 error rtn
001024' 350 00 0 17 777772      ACT01S: AOS     -6(P)                   ;Normal skip return
001025' 262 17 0 00 000005              GET     AC5                     ;Restore ac's
001026' 262 17 0 00 000004              GET     AC4
001027' 262 17 0 00 000003              GET     AC3
001030' 262 17 0 00 000002              GET     AC2
001031' 262 17 0 00 000001              GET     AC1
001032' 262 17 0 00 000000              GET     AC0
001033' 263 17 0 00 000000              RTN                             ;Error return
```

```
                                        SUBTTL  CTP Buffer Unmap

                                ;Here we unmap a buffer from sca.

001034' 332 00 0 00 000216'     UNMAPB: TRMSG   (Opcode: Buffer unmap)
001035' 260 17 0 00 261077
001036' 260 17 0 00 003543'             GO      CHGOPC                  ;Change opcode from a req to a res.
001037' 260 17 0 00 001052'             GO      VAL02                   ;Check the validity of the ctp request
001040' 254 00 0 00 001047'             JRST    UNMAP1                  ;Error in the validity check
001041' 260 17 0 00 001111'             GO      ACT02                   ;Perform action for the request
001042' 255 00 0 00 000000              JFCL                            ;Error return
001043' 661 16 0 00 000002              TLO     CURSTS,(DONE)           ;Set the done bit
001044' 621 16 0 00 000001              TLZ     CURSTS,(GOBIT)          ;Clear the gobit
001045' 200 01 0 00 000276'             MOVE    AC1,CUREXE              ;Get the rhblk
001046' 202 16 0 01 000002              MOVEM   CURSTS,STATUS(AC1)      ;Store the status
001047' 260 17 0 00 004124'     UNMAP1: GO      SEND                    ;Send the ctp response message
001050' 255 00 0 00 000000              JFCL                            ;Virtual circuit gone return
001051' 263 17 0 00 000000              RTN                             ;Return

                                ;Check the validity of the ctp request

001052' 261 17 0 00 000000      VAL02:  PUT     AC0                     ;Save ac's
001053' 261 17 0 00 000001              PUT     AC1
001054' 261 17 0 00 000002              PUT     AC2
001055' 261 17 0 00 000003              PUT     AC3
001056' 200 01 0 00 000276'             MOVE    AC1,CUREXE              ;Get rhblk
001057' 200 03 0 01 000003              MOVE    AC3,CTPPKT(AC1)         ;Now get the rblk
001060' 201 01 0 03 000001              MOVEI   AC1,1(AC3)              ;Get buftype address word
001061' 135 00 0 00 000315'             LDB     AC0,BUFT                ;Get buftype
001062' 302 00 0 00 000000              CAIE    AC0,0                   ;Value 0?
001063' 254 00 0 00 001066'             JRST    VAL02G                  ;No
001064' 201 00 0 00 000376              MOVEI   AC0,NONSEN              ;Yes, do request not understood
001065' 254 00 0 00 001101'             JRST    VAL02X                  ;Take the +1 error return
001066' 307 00 0 00 000002      VAL02G: CAIG    AC0,2                   ;Value bigger than 2?
001067' 254 00 0 00 001072'             JRST    VAL02A                  ;No, continue
001070' 201 00 0 00 000376              MOVEI   AC0,NONSEN              ;Yes, do request not understood
001071' 254 00 0 00 001101'             JRST    VAL02X                  ;Take the +1 error return

                                ;Check to see if 'BUFLENGTH'' is to big.

001072' 260 17 0 00 003450'     VAL02A: GO      LDSTUS                  ;Go load status value of 1 or 2
001073' 201 01 0 03 000003              MOVEI   AC1,3(AC3)              ;Get buflength address word
001074' 135 00 0 00 000340'             LDB     AC0,BUFL                ;Get buflength
001075' 260 17 0 00 003463'             GO      P11T10                  ;Change pdp11 word to pdp10 word
001076' 307 00 0 00 002000              CAIG    AC0,^D1024              ;Over max no of words?
001077' 254 00 0 00 001103'             JRST    VAL02S                  ;No, take the +2 return
001100' 201 00 0 00 000375              MOVEI   AC0,NORESO              ;Yes, do resource failure
001101' 260 17 0 00 003450'     VAL02X: GO      LDSTUS                  ;Load the status byte in the rblk
001102' 334 00 0 00 000000              SKIPA                           ;Take the +1 error return
001103' 350 00 0 17 777774      VAL02S: AOS     -4(P)                   ;+2 Return
001104' 262 17 0 00 000003              GET     AC3
001105' 262 17 0 00 000002              GET     AC2
001106' 262 17 0 00 000001              GET     AC1
001107' 262 17 0 00 000000              GET     AC0
001110' 263 17 0 00 000000              RTN                             ;Return
```

```
                                ;Do the function the ctp request packet demanded

001111' 261 17 0 00 000000      ACT02:  PUT     AC0                     ;Save ac's
001112' 261 17 0 00 000001              PUT     AC1
001113' 261 17 0 00 000002              PUT     AC2
001114' 261 17 0 00 000003              PUT     AC3

                                ;Get the 'BUFLNAME' from the ctp packet

001115' 200 01 0 00 000276'             MOVE    AC1,CUREXE              ;Get rhblk
001116' 200 03 0 01 000003              MOVE    AC3,CTPPKT(AC1)         ;Now get the rblk
001117' 201 01 0 03 000004              MOVEI   AC1,4(AC3)              ;Get buflname address word
001120' 135 00 0 00 000345'             LDB     AC0,BUFS                ;Get buffer name from the ctp packet
001121' 260 17 0 00 003463'             GO      P11T10                  ;Change pdp11 word to pdp10 word
001122' 242 00 0 00 000004              LSH     AC0,4                   ;Left justify it
001123' 202 00 0 00 000267'             MOVEM   AC0,SCALBF              ;Save it

                                ;See if this buffer is really mapped by looking in the cblks.

001124' 476 00 0 00 000002              SETOM   AC2                     ;Init index pointer
001125' 201 03 0 00 000040              MOVEI   AC3,CBLSTE              ;Get size of cblst
001126' 350 00 0 00 000002      ACT02A: AOS     AC2                     ;Increment index
001127' 200 04 0 02 010556'             MOVE    AC4,CBLST(AC2)          ;Get cblk address
001130' 312 00 0 04 000006              CAME    AC0,BFSCA1(AC4)         ;Is this sca buffer the same?
001131' 254 00 0 00 001136'             JRST    ACT02B                  ;No, continue
001132' 402 00 0 04 000006              SETZM   BFSCA1(AC4)             ;Yes, clear cblk location 'BFSCA1'
001133' 200 00 0 04 000007              MOVE    AC0,BUFF1(AC4)          ;Get bufx address
001134' 402 00 1 00 000000              SETZM   @AC0                    ;Make this buffer(bufx) available
001135' 254 00 0 00 001151'             JRST    ACT02E                  ;This buffer is mapped

001136' 312 00 0 04 000010      ACT02B: CAME    AC0,BFSCA2(AC4)         ;Is this sca buffer the same?
001137' 367 03 0 00 001126'             SOJG    AC3,ACT02A              ;No, keep looking
001140' 336 00 0 00 000003              SKIPN   AC3                     ;Looked in the whole table?
001141' 254 00 0 00 001146'             JRST    ACT02C                  ;Yes
001142' 402 00 0 04 000010              SETZM   BFSCA2(AC4)             ;Yes, clear cblk location 'BFSCA2'
001143' 200 00 0 04 000011              MOVE    AC0,BUFF2(AC4)          ;Get bufx address
001144' 402 00 1 00 000000              SETZM   @AC0                    ;Make this buffer(bufx) available
001145' 254 00 0 00 001151'             JRST    ACT02E                  ;This buffer is mapped
```

```
                                        ;Buffer is not mapped
                                        
001146' 201 00 0 00 000371      ACT02C: MOVEI   AC0,NOBUFM      ;Yes, do no buffer mapped
001147' 260 17 0 00 003450'             GO      LDSTUS          ;Load the status byte in the rblk
001150' 254 00 0 00 001160'             JRST    ACT02X          ;Exit, take +1 error rtn
                                        
                                        ;Unmap this buffer from sca.
                                        
001151' 200 01 0 00 000267'     ACT02E: MOVE    AC1,SCALBF      ;Get the buffer name
001152' 260 17 0 00 006040'             GO      RMVBUF          ;Unmap this buffer from sca
001153' 254 00 0 00 001157'             JRST    ACT02S          ;Good return
001154' 201 00 0 00 000375              MOVEI   AC0,NORESO      ;Error, do resource failure
001155' 260 17 0 00 003450'             GO      LDSTUS          ;Load the status byte in the rblk
001156' 254 00 0 00 001160'             JRST    ACT02X          ;Take the +1 error return
001157' 350 00 0 17 777774      ACT02S: AOS     -4(P)           ;Normal skip return
001160' 262 17 0 00 000003      ACT02X: GET     AC3             ;Restore ac's
001161' 262 17 0 00 000002              GET     AC2
001162' 262 17 0 00 000001              GET     AC1
001163' 262 17 0 00 000000              GET     AC0
001164' 263 17 0 00 000000              RTN                     ;Return
```

```
                        SUBTTL  CTP Move Buffer

                ;This routine either reads from or writes to a buffer(dma xfer) in another
                ;node.  For 3rd party transfers we should have received a "CONNECT" request
                ;from this same controller, establishing a 3rd party connection to a responder.

001165' 332 00 0 00 000216'     MOVEBF: TRMSG   (Opcode: Move buffer)
001166' 260 17 0 00 261111
001167' 261 17 0 00 000000              PUT     AC0                     ;Save ac's
001170' 261 17 0 00 000001              PUT     AC1
001171' 261 17 0 00 000002              PUT     AC2
001172' 603 16 0 00 000004              TLNE    CURSTS,(ONETIM)         ;First time here?
001173' 254 00 0 00 001201'             JRST    MOVEB1                  ;No, go around the delay
001174' 260 17 0 00 001236'             GO      VAL03                   ;Check the validity of the ctp request
001175' 254 00 0 00 001223'             JRST    MOVEB2                  ;Error in the validity check
001176' 260 17 0 00 004170'             GO      EXDLY                   ;Do the delay
001177' 254 00 0 00 001231'             JRST    MOVEB5                  ;+1  Delay has been started, go around
001200' 254 00 0 00 001223'             JRST    MOVEB2                  ;+2  "CFORK" Failure
001201' 402 00 0 00 000222'     MOVEB1: SETZM   LASTF                   ;Init the "LAST XFER FLAG"
001202' 200 02 0 00 000276'             MOVE    AC2,CUREXE              ;Get rblk address
001203' 554 01 0 02 000001              HLRZ    AC1,RPCNT(AC2)          ;Get the repeat count
001204' 336 00 0 00 000001              SKIPN   AC1                     ;Repeat count = 0
001205' 476 00 0 00 000222'             SETOM   LASTF                   ;Yes, set the "LAST XFER FLAG"

001206' 260 17 0 00 001322'     MOVEBA: GO      ACT03                   ;Perform action for the request
001207' 254 00 0 00 001223'             JRST    MOVEB2                  ; ERROR return
001210' 254 00 0 00 001230'             JRST    MOVEB4                  ; Transfer started return
001211' 600 00 0 00 000000              NOP                             ; Dma done return

001212' 200 02 0 00 000276'             MOVE    AC2,CUREXE              ;Get rblk address
001213' 554 01 0 02 000001              HLRZ    AC1,RPCNT(AC2)          ;Offset to the repcount
001214' 322 01 0 00 001223'             JUMPE   AC1,MOVEB2              ;If repeat = 0 we are all done
001215' 370 00 0 00 000001              SOS     AC1                     ;We start counting at 0 so subtract rep.
001216' 260 17 0 00 003364'             GO      GETACT                  ;Get the actcount
001217' 312 00 0 00 000001              CAME    AC0,AC1                 ;Actcount = repcount
001220' 254 00 0 00 001231'             JRST    MOVEB5                  ;No
001221' 350 00 0 00 000000              AOS     AC0                     ;Yes, increment the actcount
001222' 260 17 0 00 003402'             GO      STRACT                  ;Now store it

001223' 260 17 0 00 003543'     MOVEB2: GO      CHGOPC                  ;Change opcode from a req to a res.
001224' 476 00 0 00 000222'             SETOM   LASTF                   ;Set the "LAST XFER FLAG"
001225' 260 17 0 00 004124'             GO      SEND                    ;Send the ctp response message
001226' 255 00 0 00 000000              JFCL                            ;Virtual circuit gone return
001227' 661 16 0 00 000002              TLO     CURSTS,(DONE)           ;Set the done bit
001230' 621 16 0 00 000001      MOVEB4: TLZ     CURSTS,(GOBIT)          ;Clear the gobit
001231' 661 16 0 00 000004      MOVEB5: TLO     CURSTS,(ONETIM)         ;Set the onetim flag
001232' 262 17 0 00 000002              GET     AC2
001233' 262 17 0 00 000001              GET     AC1
001234' 262 17 0 00 000000              GET     AC0
001235' 263 17 0 00 000000              RTN                             ;Return
```

```
                                        ;Check the validity of the ctp request

001236' 261 17 0 00 000000      VAL03:  PUT     AC0                     ;Save ac's
001237' 261 17 0 00 000001              PUT     AC1
001240' 261 17 0 00 000002              PUT     AC2
001241' 261 17 0 00 000003              PUT     AC3
001242' 200 01 0 00 000276'             MOVE    AC1,CUREXE              ;Get rhblk
001243' 200 03 0 01 000003              MOVE    AC3,CTPPKT(AC1)         ;Get rblk

                                        ;Validate the "MOVETYPE" byte

001244' 201 01 0 03 000002              MOVEI   AC1,2(AC3)              ;Get "MOVETYPE" address word
001245' 135 00 0 00 000324'             LDB     AC0,MOVE                ;Get movetype
001246' 303 00 0 00 000003              CAILE   AC0,3                   ;Value over max?
001247' 254 00 0 00 001302'             JRST    VAL03G                  ;Yes, do request not understood
001250' 307 00 0 00 000001              CAIG    AC0,1                   ;Asking for maintenance?
001251' 254 00 0 00 001254'             JRST    VAL03A                  ;No, continue
001252' 201 00 0 00 000377              MOVEI   AC0,NTIMPL              ;Do function not implemented
001253' 254 00 0 00 001312'             JRST    VAL03X                  ;Do +1 error return

                                        ;Validate the "PKTSIZ" byte

001254' 201 01 0 03 000002      VAL03A: MOVEI   AC1,2(AC3)              ;Get "PKTSIZ" address word
001255' 135 00 0 00 000333'             LDB     AC0,PKTZ                ;Get pktsiz
001256' 307 00 0 00 000001              CAIG    AC0,1                   ;Value over max?
001257' 254 00 0 00 001261'             JRST    VAL03B                  ;No, proceed
001260' 254 00 0 00 001302'             JRST    VAL03G                  ;Request not understood

                                        ;Validate the "OTHERNODE" byte

001261' 201 01 0 03 000002      VAL03B: MOVEI   AC1,2(AC3)              ;Get "OTHERNODE" address word
001262' 135 00 0 00 000331'             LDB     AC0,OTHE                ;Get othernode
001263' 335 00 0 00 000000              SKIPGE  AC0                     ;Value a negative number?
001264' 254 00 0 00 001302'             JRST    VAL03G                  ;Yes, request not understood

                                        ;Check to see if "BUFLENGTH" is to big.

001265' 201 01 0 03 000003      VAL03C: MOVEI   AC1,3(AC3)              ;Get "BUFLENGTH" address word
001266' 135 00 0 00 000340'             LDB     AC0,BUFL                ;Get buflength
001267' 260 17 0 00 003463'             GO      P11T10                  ;Change pdp11 word to pdp10 word
001270' 202 00 0 00 000227'             MOVEM   AC0,BUFLEN              ;Store it
001271' 307 00 0 00 002000              CAIG    AC0,^D1024              ;Over max no of words?
001272' 254 00 0 00 001275'             JRST    VAL03F                  ;No, proceed
001273' 201 00 0 00 000375              MOVEI   AC0,NORESO              ;Yes, do resource failure
001274' 254 00 0 00 001312'             JRST    VAL03X                  ;Take the +1 error return
```

```
1024
1025                                    ;Check and see if the buffer offset(buflofset) is to big.
1026
1027  001275' 201 01 0 03 000005    VAL03F: MOVEI   AC1,5(AC3)              ;Get "BUFLOFSET" address word
1028  001276' 135 00 0 00 000347'           LDB     AC0,BUFO                ;Get buflofset
1029  001277' 260 17 0 00 003463'           GO      P11T10                  ;Change pdp11 word to pdp10 word
1030  001300' 317 00 0 00 000227'           CAMG    AC0,BUFLEN              ;Offset greater than the buffer length?
1031  001301' 254 00 0 00 001304'           JRST    VAL03H                  ;No, proceed
1032  001302' 201 00 0 00 000376    VAL03G: MOVEI   AC0,NONSEN              ;Yes, do request not understood
1033  001303' 254 00 0 00 001312'           JRST    VAL03X                  ;Take the +1 error return
1034
1035                                    ;Get repcount and store it
1036
1037  001304' 260 17 0 00 003364'   VAL03H: GO      GETREP                  ;Get repcount and
1038  001305' 200 02 0 00 000276'           MOVE    AC2,CUREXE              ; Store
1039  001306' 506 00 0 02 000001            HRLM    AC0,RPCNT(AC2)          ; It
1040
1041                                    ;Init the actcount field. It should return value of 1 first to conform
1042                                    ; to the way the HSC does it.
1043
1044  001307' 402 00 0 00 000000            SETZM   AC0                     ;Init actcount to 0
1045  001310' 260 17 0 00 003402'           GO      STRACT                  ; For the first time
1046  001311' 254 00 0 00 001314'           JRST    VAL03S                  ;Take the good return
1047
1048  001312' 260 17 0 00 003450'   VAL03X: GO      LDSTUS                  ;Load the status byte in the rblk
1049  001313' 334 00 0 00 000000            SKIPA                           ;Take the +1 error return
1050  001314' 350 00 0 17 777774    VAL03S: AOS     -4(P)                   ;+2 Return
1051  001315' 262 17 0 00 000003            GET     AC3
1052  001316' 262 17 0 00 000002            GET     AC2
1053  001317' 262 17 0 00 000001            GET     AC1
1054  001320' 262 17 0 00 000000            GET     AC0
1055  001321' 263 17 0 00 000000            RTN                             ;Return
1056
1057
```

```
                                        ;Do the function the ctp request packet demanded
                                        
001322' 261 17 0 00 000000      ACT03:  PUT     AC0                     ;Save ac's
001323' 261 17 0 00 000001              PUT     AC1
001324' 261 17 0 00 000002              PUT     AC2
001325' 261 17 0 00 000003              PUT     AC3
001326' 261 17 0 00 000004              PUT     AC4
001327' 261 17 0 00 000005              PUT     AC5
                                        
                                        ;Set up the status byte for success.
                                        
001330' 201 00 0 00 000000              MOVEI   AC0,SUCCES              ;Load successful status
001331' 260 17 0 00 003450'             GO      LDSTUS                  ;Do it
                                        
                                        ;Get the controllers ci number and store it.
                                        
001332' 550 01 0 00 000016              HRRZ    AC1,CURSTS              ;Get the cblk in ac1
001333' 200 01 0 01 000005              MOVE    AC1,LOCPRT(AC1)         ;Now get the local port number(ci #)
001334' 202 01 0 00 000256'             MOVEM   AC1,CIPORT              ; And store it
                                        
                                        ;Get movetype and store it
                                        
001335' 200 01 0 00 000276'     ACT03A: MOVE    AC1,CUREXE              ;Get rhblk
001336' 200 03 0 01 000003              MOVE    AC3,CTPPKT(AC1)         ;Now get the rblk
001337' 201 01 0 03 000002              MOVEI   AC1,2(AC3)              ;Get movetype address word
001340' 135 00 0 00 000324'             LDB     AC0,MOVE                ;Get genfunct
001341' 202 00 0 00 000233'             MOVEM   AC0,MOVETY              ;Store it
                                        
                                        ;Get othernode and store it
                                        
001342' 135 00 0 00 000331'             LDB     AC0,OTHE                ;Get othernode
001343' 202 00 0 00 000232'             MOVEM   AC0,OTHERN              ;Store it
                                        
                                        ;Check and see if we have the same local buffer mapped(buflname).
                                        
001344' 201 01 0 03 000004              MOVEI   AC1,4(AC3)              ;Get 'BUFLNAME" address word
001345' 135 00 0 00 000345'             LDB     AC0,BUFS                ;Get buflname
001346' 260 17 0 00 003463'             GO      P11T10                  ;Change pdp11 word to pdp10 word
001347' 242 00 0 00 000004              LSH     AC0,4                   ;Left justify it
001350' 202 00 0 00 000267'             MOVEM   AC0,SCALBF              ;Store buffer name
001351' 336 00 0 00 000267'             SKIPN   SCALBF                  ;Skip if non zero buffer name
001352' 254 00 0 00 001363'             JRST    ACT3B1                  ; It was 0
                                        
001353' 476 00 0 00 000002              SETOM   AC2                     ;Init index pointer
001354' 201 03 0 00 000040              MOVEI   AC3,CBLSTE              ;Get size of cblst
001355' 350 00 0 00 000002      ACT03B: AOS     AC2                     ;Increment index
001356' 200 04 0 02 010556'             MOVE    AC4,CBLST(AC2)          ;Get cblk address
001357' 312 00 0 04 000006              CAME    AC0,BFSCA1(AC4)         ;Is this sca buffer the same?
001360' 316 00 0 04 000010              CAMN    AC0,BFSCA2(AC4)         ;No, is this sca buffer the same?
001361' 254 00 0 00 001365'             JRST    ACT03C                  ;Yes, this buffer is mapped
001362' 367 03 0 00 001355'             SOJG    AC3,ACT03B              ;No, keep looking
```

```
                                        ;Buffer is not mapped
                                        
001363' 201 00 0 00 000371      ACT3B1: MOVEI   AC0,NOBUFM              ;Yes, do no buffer mapped
001364' 254 00 0 00 001464'             JRST    ACT03X                  ;Exit, take +1 error rtn
                                        
                                        ;See if we have a connection open to the other node specified in the ctp
                                        ;packet.  It could be the contollers node or a 3rd party node.
                                        
001365' 476 00 0 00 000002      ACT03C: SETOM   AC2                     ;Init index pointer
001366' 201 03 0 00 000040              MOVEI   AC3,CBLSTE              ;Get size of cblst
001367' 350 00 0 00 000002      ACT03D: AOS     AC2                     ;Increment index
001370' 200 04 0 02 010556'             MOVE    AC4,CBLST(AC2)          ;Get cblk address
001371' 336 00 0 04 000000              SKIPN   CID(AC4)                ;Is this cblk being used?
001372' 254 00 0 00 001401'             JRST    ACT03E                  ;No, look for another
001373' 550 01 0 04 000005              HRRZ    AC1,LOCPRT(AC4)         ;Get the controllers ci number
001374' 312 01 0 00 000256'             CAME    AC1,CIPORT              ;Is this 3rd party on the same ci?
001375' 254 00 0 00 001401'             JRST    ACT03E                  ;No
001376' 550 01 0 04 000004              HRRZ    AC1,NODE(AC4)           ;Get this cblks 3rd party node #
001377' 316 01 0 00 000232'             CAMN    AC1,OTHERN              ;Do we have a connection open?
001400' 254 00 0 00 001404'             JRST    ACT03F                  ;Yes, continue
001401' 367 03 0 00 001367'     ACT03E: SOJG    AC3,ACT03D              ;No, keep looking
001402' 201 00 0 00 000366              MOVEI   AC0,NOCONN              ;Yes, do no connection exist
001403' 254 00 0 00 001464'             JRST    ACT03X                  ;Exit, take +1 error rtn
                                        
                                        ;See if the last read or write we did was completed
                                        
001404' 607 16 0 00 000100      ACT03F: TLNN    CURSTS,(WAITT)          ;Waiting for a move(r/w) to complete?
001405' 254 0C 0 00 001417'             JRST    ACT03I                  ;No
001406' 200 00 0 04 000001              MOVE    AC0,CBSTS(AC4)          ;Yes, get the cblk status
001407' 607 00 0 00 040000              TLNN    AC0,(DMA)               ;Has the xfer(read or write) completed?
001410' 254 00 0 00 001415'             JRST    ACT03G                  ;No, error
001411' 621 00 0 00 040000              TLZ     AC0,(DMA)               ;Yes, clear the "DMA" flag
001412' 202 00 0 04 000001              MOVEM   AC0,CBSTS(AC4)          ;Update the cblk status
001413' 621 16 0 00 000100              TLZ     CURSTS,(WAITT)          ;Yes, clear the "WAITT" flag
001414' 254 00 0 00 001466'             JRST    ACT03Z                  ;Take the +2 good return
001415' 201 00 0 00 000373      ACT03G: MOVEI   AC0,FUNCFL              ;Yes, do no function failure
001416' 254 00 0 00 001464'             JRST    ACT03X                  ;Exit, take +1 error rtn
```

```
1147
1148                                    ;Increment the actcount field
1149
1150  001417' 260 17 0 00 003364'   ACT03I: GO      GETACT                  ;Get actcount
1151  001420' 350 00 0 00 000000            AOS     AC0                     ;Increment it
1152  001421' 260 17 0 00 003402'            GO      STRACT                  ;Now store it
1153
1154                                    ;Now we are ready to send or receive data.
1155
1156  001422' 200 01 0 00 000276'            MOVE    AC1,CUREXE              ;Get rhblk
1157  001423' 200 03 0 01 000003            MOVE    AC3,CTPPKT(AC1)         ;Now get the rblk
1158  001424' 201 01 0 03 000006            MOVEI   AC1,6(AC3)              ;Get remote buffer address word
1159  001425' 135 00 0 00 000351'            LDB     AC0,BUFD                ;Get remote buffer (bytes 24-27)
1160  001426' 260 17 0 00 003463'            GO      P11T10                  ;Convert to pdp10 word
1161  001427' 242 00 0 00 000004            LSH     AC0,4                   ;Left justify it
1162  001430' 202 00 0 00 000270'            MOVEM   AC0,SCARBF              ;Store it
1163
1164  001431' 201 01 0 03 000005            MOVEI   AC1,5(AC3)              ;Get local buffer offset address word
1165  001432' 135 00 0 00 000347'            LDB     AC0,BUFO                ;Get local buffer offset (bytes 20-23)
1166  001433' 260 17 0 00 003463'            GO      P11T10                  ;Convert to pdp10 word
1167  001434' 202 00 0 00 000271'            MOVEM   AC0,LBUFOF              ;Store it
1168
1169  001435' 201 01 0 03 000007            MOVEI   AC1,7(AC3)              ;Get remote buffer offset address word
1170  001436' 135 00 0 00 000353'            LDB     AC0,BUDO                ;Get remote buffer offset (bytes 28-31)
1171  001437' 260 17 0 00 003463'            GO      P11T10                  ;Convert to pdp10 word
1172  001440' 202 00 0 00 000272'            MOVEM   AC0,RBUFOF              ;Store it
1173
1174  001441' 550 01 0 00 000016            HRRZ    AC1,CURSTS              ;Get cblkn
1175  001442' 200 01 0 01 000000            MOVE    AC1,CID(AC1)            ;Get the cid
1176
1177                                    ;Do either a read or a write.
1178
1179  001443' 200 00 0 00 000233'            MOVE    AC0,MOVETY              ;Get the movetype byte
1180  001444' 306 00 0 00 000001            CAIN    AC0,1                   ;Do a write?
1181  001445' 254 00 0 00 001455'            JRST    ACT03J                  ;Yes
```

```
1182
1183                                        ;Here we do the read.
1184
1185  001446' 200 02 0 00 000270'           MOVE    AC2,SCARBF          ;Get remote buffer name
1186  001447' 200 03 0 00 000267'           MOVE    AC3,SCALBF          ;Get local buffer name
1187  001450' 514 04 0 00 000272'           HRLZ    AC4,RBUFOF          ;Get remote buffer offset
1188  001451' 540 04 0 00 000271'           HRR     AC4,LBUFOF          ;Get local buffer offset
1189  001452' 260 17 0 00 006115'           GO      REDATA              ;Do the read
1190  001453' 661 16 0 00 000100            TLO     CURSTS,(WAITT)      ;Set 'WAITING FOR A MOVE(R/W)' flag
1191  001454' 254 00 0 00 001467'           JRST    ACT03S              ;Do +2 good return
1192
1193                                        ;Here we do the write.
1194
1195  001455' 200 02 0 00 000267'  ACT03J:  MOVE    AC2,SCALBF          ;Get local buffer name
1196  001456' 200 03 0 00 000270'           MOVE    AC3,SCARBF          ;Get remote buffer name
1197  001457' 514 04 0 00 000271'           HRLZ    AC4,LBUFOF          ;Get local buffer offset
1198  001460' 540 04 0 00 000272'           HRR     AC4,RBUFOF          ;Get remote buffer offset
1199  001461' 260 17 0 00 006066'           GO      SEDATA              ;Do the write
1200  001462' 661 16 0 00 000100            TLO     CURSTS,(WAITT)      ;Set 'WAITING FOR A MOVE(R/W)' flag
1201  001463' 254 00 0 00 001467'           JRST    ACT03S              ;Do +2 good return
1202
1203  001464' 260 17 0 00 003450'  ACT03X:  GO      LDSTUS              ;Load the status byte in the rblk
1204  001465' 254 00 0 00 001470'           JRST    ACT03T              ;Do +1 error return
1205
1206  001466' 350 00 0 17 777772   ACT03Z:  AOS     -6(P)               ;Here for DMA done skip return
1207  001467' 350 00 0 17 777772   ACT03S:  AOS     -6(P)               ;Here for transfer has started skip
1208                                                                    ; return
1209  001470' 262 17 0 00 000005   ACT03T:  GET     AC5                 ;Restore ac's
1210  001471' 262 17 0 00 000004            GET     AC4
1211  001472' 262 17 0 00 000003            GET     AC3
1212  001473' 262 17 0 00 000002            GET     AC2
1213  001474' 262 17 0 00 000001            GET     AC1
1214  001475' 262 17 0 00 000000            GET     AC0
1215  001476' 263 17 0 00 000000            RTN                         ;Error return
```

```
                                        SUBTTL  CTP Generate Message and Datagram

                                ;This routine will send back to the controller 1 or more messages or datagrams.

001477'                         GENDGM:
001477' 332 00 0 00 000216'     GENMSG: TRMSG   (Opcode: Generate Message/Datagram)
001500' 260 17 0 00 261126
001501' 261 17 0 00 000000              PUT     AC0                     ;Save ac's
001502' 261 17 0 00 000001              PUT     AC1
001503' 261 17 0 00 000002              PUT     AC2
001504' 603 16 0 00 000004              TLNE    CURSTS,(ONETIM)         ;First time here?
001505' 254 00 0 00 001514'             JRST    GENMS1                  ;No, go around the delay
001506' 260 17 0 00 003543'             GO      CHGOPC                  ;Change opcode from a req to a res.
001507' 260 17 0 00 001552'             GO      VAL04                   ;Check the validity of the ctp request
001510' 254 00 0 00 001540'             JRST    GENMS2                  ;Error in the validity check
001511' 260 17 0 00 004170'             GO      EXDLY                   ;Do the delay
001512' 254 00 0 00 001545'             JRST    GENMS5                  ;+1  Delay has been started, go around
001513' 254 00 0 00 001540'             JRST    GENMS2                  ;+2  "CFORK" Failure
001514' 402 00 0 00 000222'     GENMS1: SETZM   LASTF                   ;Init the "LAST XFER FLAG"
001515' 200 02 0 00 000276'             MOVE    AC2,CUREXE              ;Get rblk address
001516' 554 01 0 02 000001              HLRZ    AC1,RPCNT(AC2)          ;Get the repeat count
001517' 326 01 0 00 001523'             JUMPN   AC1,GENMS6              ;Jump if repeat count = 0
001520' 603 16 0 00 000400              TLNE    CURSTS,(DGMSG)          ;Yes, check dgmsg bit
001521' 621 16 0 00 000420              TLZ     CURSTS,(DGM!DGMSG)      ;Clr dgm and dgmsg bits
001522' 476 00 0 00 000222'             SETOM   LASTF                   ;Yes, set the "LAST XFER FLAG"
001523' 260 17 0 00 001700'     GENMS6: GO      ACT04                   ;Perform action for the request
001524' 254 00 0 00 001540'             JRST    GENMS2                  ;Error return
001525' 200 02 0 00 000276'             MOVE    AC2,CUREXE              ;Get rblk address
001526' 554 01 0 02 000001              HLRZ    AC1,RPCNT(AC2)          ;Offset to the repcount
001527' 322 01 0 00 001543'             JUMPE   AC1,GENMS4              ;If repeat = 0 we are all done
001530' 370 00 0 00 000001              SOS     AC1                     ;We start counting at 0 so subtract rep.
001531' 260 17 0 00 003364'             GO      GETACT                  ;Get the actcount
001532' 312 00 0 00 000001              CAME    AC0,AC1                 ;Actcount = repcount
001533' 254 00 0 00 001545'             JRST    GENMS5                  ;No
001534' 350 00 0 00 000000              AOS     AC0                     ;Yes, increment the actcount
001535' 260 17 0 00 003402'             GO      STRACT                  ;Now store it
001536' 603 16 0 00 000400              TLNE    CURSTS,(DGMSG)          ;Check dgmsg bit
001537' 621 16 0 00 000420              TLZ     CURSTS,(DGM!DGMSG)      ;Clr dgm and dgmsg bits
001540' 476 00 0 00 000222'     GENMS2: SETOM   LASTF                   ;Set the "LAST XFER FLAG"
001541' 260 17 0 00 004147'             GO      SEN04                   ;Send the ctp response message
001542' 255 00 0 00 000000              JFCL                            ;Virtual circuit gone return
001543' 621 16 0 00 000001      GENMS4: TLZ     CURSTS,(GOBIT)          ;Clear the gobit
001544' 661 16 0 00 000002              TLO     CURSTS,(DONE)           ;Set the done bit

001545' 661 16 0 00 000004      GENMS5: TLO     CURSTS,(ONETIM)         ;Set the onetim flag
001546' 262 17 0 00 000002              GET     AC2
001547' 262 17 0 00 000001              GET     AC1
001550' 262 17 0 00 000000              GET     AC0
001551' 263 17 0 00 000000              RTN                             ;Return
```

```
1266                                    ;Check the validity of the ctp request
1267
1268  001552' 261 17 0 00 000000  VAL04:  PUT     AC0                     ;Save ac's
1269  001553' 261 17 0 00 000001          PUT     AC1
1270  001554' 261 17 0 00 000002          PUT     AC2
1271  001555' 261 17 0 00 000003          PUT     AC3
1272  001556' 261 17 0 00 000004          PUT     AC4
1273  001557' 261 17 0 00 000005          PUT     AC5
1274  001560' 200 01 0 00 000276'         MOVE    AC1,CUREXE              ;Get rhblk
1275  001561' 200 02 0 01 000003          MOVE    AC2,CTPPKT(AC1)         ;Now get the rblk
1276  001562' 201 01 0 02 000002          MOVEI   AC1,2(AC2)              ;Get genfunct address word
1277  001563' 135 00 0 00 000323'         LDB     AC0,GENF                ;Get genfunct
1278  001564' 307 00 0 00 000003          CAIG    AC0,3                   ;Value bigger than 3?
1279  001565' 254 00 0 00 001570'         JRST    VAL04A                  ;No, continue
1280  001566' 201 00 0 00 000376          MOVEI   AC0,NONSEN              ;Yes, do request not understood
1281  001567' 254 00 0 00 001666'         JRST    VAL04X                  ;Take the +1 error return
1282
1283                                    ;Check to see if "GENLENGTH" is to big.
1284
1285  001570' 260 17 0 00 003416' VAL04A: GO      GETGLN                  ;Get the corrected "GENLENGTH"
1286  001571' 202 00 0 00 000226'         MOVEM   AC0,GENLEN              ;Store it
1287  001572' 607 16 0 00 000020          TLNN    CURSTS,(DGM)            ;Received as a dgm?
1288  001573' 254 00 0 00 001577'         JRST    VAL04B                  ;No, go ahead
1289  001574' 317 00 0 00 000266'         CAMG    AC0,MAXDGM              ;Over max no of bytes for dgm scs% jsys?
1290  001575' 254 00 0 00 001603'         JRST    VAL04D                  ;No,
1291  001576' 254 00 0 00 001601'         JRST    VAL04C                  ;Yes
1292
1293  001577' 317 00 0 00 000265' VAL04B: CAMG    AC0,MAXMSG              ;Over max no of bytes for msg scs jsys?
1294  001600' 254 00 0 00 001603'         JRST    VAL04D                  ;No, get the repeat count
1295
1296  001601' 201 00 0 00 000375  VAL04C: MOVEI   AC0,NORESO              ;Yes, do resource failure
1297  001602' 254 00 0 00 001666'         JRST    VAL04X
1298
1299                                    ;Get repcount and store it
1300
1301  001603' 260 17 0 00 003364' VAL04D: GO      GETREP                  ;Get repcount and
1302  001604' 200 02 0 00 000276'         MOVE    AC2,CUREXE              ; Store
1303  001605' 506 00 0 02 000001          HRLM    AC0,RPCNT(AC2)          ; It
1304
1305                                    ;Set up the function the ctp request packet demanded
1306
1307                                    ;Get genfunct and store it
1308
1309  001606' 200 01 0 00 000276'         MOVE    AC1,CUREXE              ;Get rhblk
1310  001607' 200 02 0 01 000003          MOVE    AC2,CTPPKT(AC1)         ;Now get the rblk
1311  001610' 201 01 0 02 000002          MOVEI   AC1,2(AC2)              ;Get genfunct address word
1312  001611' 135 00 0 00 000323'         LDB     AC0,GENF                ;Get genfunct
1313  001612' 202 00 0 00 000224'         MOVEM   AC0,GENFUN              ;Store it
```

```
1314
1315                                    ;Get genconst and store it
1316
1317  001613' 135 00 0 00 000327'               LDB     AC0,GENC                ;Get genconst
1318  001614' 202 00 0 00 000225'               MOVEM   AC0,GENCON              ;Store it
1319
1320                                    ;Get genlength and store it
1321
1322  001615' 260 17 0 00 003416'               GO      GETGLN                  ;Get genlength
1323  001616' 202 00 0 00 000226'               MOVEM   AC0,GENLEN              ;Store it
1324
1325                                    ;Here we fill the imagedata with one of the following: a constant, byte pairs,
1326                                    ; The responder node number or the data already supplied in the packet.
1327
1328                                    ;Get the start address of the imagedata word(byte 14) in ac4
1329
1330  001617' 200 04 0 00 000276'               MOVE    AC4,CUREXE              ;Get rhblk
1331  001620' 200 02 0 04 000003                MOVE    AC2,CTPPKT(AC4)         ;Now get the rblk
1332  001621' 201 04 0 02 000003                MOVEI   AC4,3(AC2)              ;Get "IMAGEDATA" address word
1333
1334  001622' 201 00 0 00 000000                MOVEI   AC0,0                   ;Get a value
1335  001623' 312 00 0 00 000224'               CAME    AC0,GENFUN              ;Fill buffer with constant?
1336  001624' 254 00 0 00 001632'               JRST    VAL04E                  ;No
1337
1338                                    ;Fill buffer with constant
1339
1340  001625' 200 03 0 00 000226'               MOVE    AC3,GENLEN              ;Yes, get size of imagedata in bytes
1341  001626' 200 00 0 00 000225'               MOVE    AC0,GENCON              ;Get the fill constant
1342  001627' 200 02 0 00 261133                MOVE    AC2,[POINT 8,(AC4),15]  ;Init pointer
1343  001630' 136 00 0 00 000002                IDPB    AC0,AC2                 ;Deposit the fill constant in the byte
1344  001631' 367 03 0 00 001630'               SOJG    AC3,.-1                 ;Fill up the buffer
1345  001632' 201 00 0 00 000001    VAL04E:     MOVEI   AC0,1                   ;Get a value
1346  001633' 312 00 0 00 000224'               CAME    AC0,GENFUN              ;Fill buffer with byte pair numbers?
1347  001634' 254 00 0 00 001646'               JRST    VAL04G                  ;No
1348
1349                                    ;Fill buffer with byte pairs
1350
1351  001635' 200 03 0 00 000226'               MOVE    AC3,GENLEN              ;Yes, get size of imagedata in bytes
1352  001636' 201 05 0 00 000000                MOVEI   AC5,0                   ;Get the 1st byte pair number
1353  001637' 200 02 0 00 261133                MOVE    AC2,[POINT 8,(AC4),15]  ;Init pointer
1354  001640' 200 00 0 00 000005    VAL04F:     MOVE    AC0,AC5                 ;Get 1st byte pair
1355  001641' 136 00 0 00 000002                IDPB    AC0,AC2                 ;Deposit the 1st byte
1356  001642' 242 00 0 00 777770                LSH     AC0,-^D8                ;Go to 2nd byte
1357  001643' 136 00 0 00 000002                IDPB    AC0,AC2                 ;Deposit the 2nd byte
1358  001644' 350 00 0 00 000005                AOS     AC5                     ;Increment byte pair number
1359  001645' 367 03 0 00 001640'               SOJG    AC3,VAL04F              ;Fill up the buffer
1360  001646' 201 00 0 00 000002    VAL04G:     MOVEI   AC0,2                   ;Get a value
1361  001647' 312 00 0 00 000224'               CAME    AC0,GENFUN              ;Fill buffer with responder node number?
1362  001650' 254 00 0 00 001656'               JRST    VAL04H                  ;No
```

```
1363
1364                                    ;Fill buffer with the responder node number
1365
1366  001651' 200 03 0 00 000226'               MOVE    AC3,GENLEN              ;Yes, get size of imagedata in bytes
1367  001652' 200 00 0 00 000247'               MOVE    AC0,OURNOD              ;Get our node number
1368  001653' 200 02 0 00 261133                MOVE    AC2,[POINT 8,(AC4),15]  ;Init pointer
1369  001654' 136 00 0 00 000002                IDPB    AC0,AC2                 ;Deposit the node number in the byte
1370  001655' 367 03 0 00 001654'               SOJG    AC3,.-1                 ;Fill up the buffer
1371
1372                                    ;This is where we build the remaining byte fields in the ctp response packet
1373
1374  001656' 200 01 0 00 000276'  VAL04H: MOVE    AC1,CUREXE              ;Get rhblk
1375  001657' 200 02 0 01 000003                MOVE    AC2,CTPPKT(AC1)         ;Now get the rblk
1376  001660' 201 01 0 02 000002                MOVEI   AC1,2(AC2)              ;Get genconst address word
1377  001661' 201 00 0 00 000000                MOVEI   AC0,0                   ;Clear the genconst byte making
1378  001662' 137 00 0 00 000327'               DPB     AC0,GENC                ; It the portstat byte
1379
1380                                    ;Initialize the actcount field
1381
1382  001663' 476 00 0 00 000000                SETOM   AC0                     ;Init actcount to -1
1383  001664' 260 17 0 00 003402'               GO      STRACT                  ; For the first time
1384  001665' 254 00 0 00 001670'               JRST    VAL04S                  ;Take +2 return
1385
1386  001666' 260 17 0 00 003450'  VAL04X: GO      LDSTUS                  ;Load the status byte in the rblk
1387  001667' 334 00 0 00 000000                SKIPA                           ;Take the +1 error return
1388  001670' 350 00 0 17 777772   VAL04S: AOS     -6(P)                   ;+2 Return
1389  001671' 262 17 0 00 000005                GET     AC5
1390  001672' 262 17 0 00 000004                GET     AC4
1391  001673' 262 17 0 00 000003                GET     AC3
1392  001674' 262 17 0 00 000002                GET     AC2
1393  001675' 262 17 0 00 000001                GET     AC1
1394  001676' 262 17 0 00 000000                GET     AC0
1395  001677' 263 17 0 00 000000                RTN                             ;Return
```

```
1396                                    ;Do the function the ctp request packet demanded
1397
1398  001700' 261 17 0 00 000000  ACT04:  PUT     AC0                     ;Save ac's
1399  001701' 261 17 0 00 000001          PUT     AC1
1400  001702' 261 17 0 00 000002          PUT     AC2
1401
1402                                    ;Set up the status byte for success.
1403
1404  001703' 201 00 0 00 000000          MOVEI   AC0,SUCCES              ;Load successful status
1405  001704' 260 17 0 00 003450'         GO      LDSTUS                  ;Load the status byte in the rblk
1406
1407                                    ;Update actcount field
1408
1409  001705' 260 17 0 00 003364'         GO      GETACT                  ;Get actcount
1410  001706' 350 00 0 00 000000          AOS     AC0                     ;Increment it
1411  001707' 260 17 0 00 003402'         GO      STRACT                  ;Now store it
1412
1413                                    ;Now we are ready to send the message
1414
1415  001710' 260 17 0 00 004147'         GO      SEN04                   ;Send a message
1416  001711' 254 00 0 00 001713'         JRST    ACT04X                  ;+1 Error return
1417  001712' 350 00 0 17 777775  ACT04S: AOS     -3(P)                   ;Normal skip return
1418  001713' 262 17 0 00 000002  ACT04X: GET     AC2                     ;Restore ac's
1419  001714' 262 17 0 00 000001          GET     AC1
1420  001715' 262 17 0 00 000000          GET     AC0
1421  001716' 263 17 0 00 000000          RTN                             ;Error return
```

```
                        SUBTTL  CTP Generate Reset

                ;This routine will generate a reset packet on the the ci.  Before we get here
                ;we should have received a "CONNECT" request from the cint, establishing a 3rd
                ;party connection to a responder.  This routine will only be used with a cint.

001717' 332 00 0 00 000216'     GENRST: TRMSG   (Opcode: Generate reset)
001720' 260 17 0 00 261141
001721' 261 17 0 00 000000              PUT     AC0                     ;Save ac's
001722' 261 17 0 00 000001              PUT     AC1
001723' 261 17 0 00 000002              PUT     AC2
001724' 603 16 0 00 000004              TLNE    CURSTS,(ONETIM)         ;First time here?
001725' 254 00 0 00 001734'             JRST    GENRS1                  ;No, go around the delay
001726' 260 17 0 00 003543'             GO      CHGOPC                  ;Change opcode from a req to a res.
001727' 260 17 0 00 001757'             GO      VAL06                   ;Check the validity of the ctp request
001730' 254 00 0 00 001750'             JRST    GENRS2                  ;Error in the validity check
001731' 260 17 0 00 004170'             GO      EXDLY                   ;Do the delay
001732' 254 00 0 00 001752'             JRST    GENRS3                  ;+1  Delay has been started, go around
001733' 254 00 0 00 001750'             JRST    GENRS2                  ;+2  "CFORK" Failure
001734' 260 17 0 00 002006'     GENRS1: GO      ACT06                   ;Perform action for the request
001735' 254 00 0 00 001750'             JRST    GENRS2                  ;Error return
001736' 200 02 0 00 000276'             MOVE    AC2,CUREXE              ;Get rblk address
001737' 554 01 0 02 000001              HLRZ    AC1,RPCNT(AC2)          ;Offset to the repcount
001740' 370 00 0 00 000001              SOS     AC1                     ;We start counting at 0 so subtract rep.
001741' 260 17 0 00 003364'             GO      GETACT                  ;Get the actcount
001742' 312 00 0 00 000001              CAME    AC0,AC1                 ;Actcount = repcount
001743' 254 00 0 00 001752'             JRST    GENRS3                  ;No
001744' 661 16 0 00 000002              TLO     CURSTS,(DONE)           ;Set the done bit
001745' 621 16 0 00 000001              TLZ     CURSTS,(GOBIT)          ;Clear the gobit
001746' 200 01 0 00 000276'             MOVE    AC1,CUREXE              ;Get the rhblk
001747' 202 16 0 01 000002              MOVEM   CURSTS,STATUS(AC1)      ;Store the status
001750' 260 17 0 00 004124'     GENRS2: GO      SEND                    ;Send the ctp response message
001751' 255 00 0 00 000000              JFCL                            ;Virtual circuit gone return
001752' 661 16 0 00 000004      GENRS3: TLO     CURSTS,(ONETIM)         ;Set the onetim flag
001753' 262 17 0 00 000002              GET     AC2
001754' 262 17 0 00 000001              GET     AC1
001755' 262 17 0 00 000000              GET     AC0
001756' 263 17 0 00 000000              RTN                             ;Return

1460
```

```
                                        ;Check the validity of the ctp request

001757' 261 17 0 00 000000      VAL06:  PUT     AC0                     ;Save ac's
001760' 261 17 0 00 000001              PUT     AC1
001761' 261 17 0 00 000002              PUT     AC2
001762' 200 01 0 00 000276'             MOVE    AC1,CUREXE              ;Get rhblk
001763' 200 02 0 01 000003              MOVE    AC2,CTPPKT(AC1)         ;Now get the rblk
001764' 201 01 0 02 000002              MOVEI   AC1,2(AC2)              ;Get extend address word
001765' 135 00 0 00 000322'             LDB     AC0,EXTE                ;Get extend
001766' 307 00 0 00 000001              CAIG    AC0,1                   ;Value bigger than 1?
001767' 254 00 0 00 001772'             JRST    VAL06A                  ;No, continue
001770' 201 00 0 00 000376              MOVEI   AC0,NONSEN              ;Yes, do request not understood
001771' 254 00 0 00 001777'             JRST    VAL06X                  ;Take the +1 error return

                                        ;Validate the ''OTHERNODE'' byte

001772' 201 01 0 03 000002      VAL06A: MOVEI   AC1,2(AC3)              ;Get ''OTHERNODE'' address word
001773' 135 00 0 00 000331'             LDB     AC0,OTHE                ;Get othernode
001774' 331 00 0 00 000000              SKIPL   AC0                     ;Value a negative number?
001775' 254 00 0 00 002001'             JRST    VAL06S                  ;No, continue
001776' 201 00 0 00 000376              MOVEI   AC0,NONSEN              ;Yes, do request not understood

001777' 260 17 0 00 003450'     VAL06X: GO      LDSTUS                  ;Load the status byte in the rblk
002000' 334 00 0 00 000000              SKIPA                           ;Take the +1 error return
002001' 350 00 0 17 777775      VAL06S: AOS     -3(P)                   ;+2 Return
002002' 262 17 0 00 000002              GET     AC2
002003' 262 17 0 00 000001              GET     AC1
002004' 262 17 0 00 000000              GET     AC0
002005' 263 17 0 00 000000              RTN                             ;Return
```

```
                                ;Do the function the ctp request packet demanded
                                
002006' 261 17 0 00 000000      ACT06:  PUT     AC0                     ;Save ac's
002007' 261 17 0 00 000001              PUT     AC1
002010' 261 17 0 00 000002              PUT     AC2
002011' 261 17 0 00 000003              PUT     AC3
002012' 261 17 0 00 000004              PUT     AC4
002013' 261 17 0 00 000005              PUT     AC5
                                
                                ;Get the controllers ci number and store it.
                                
002014' 550 01 0 00 000016              HRRZ    AC1,CURSTS              ;Get the cblk in ac1
002015' 200 01 0 01 000005              MOVE    AC1,LOCPRT(AC1)         ;Now get the local port number(ci #)
002016' 202 01 0 00 000256'             MOVEM   AC1,CIPORT              ; And store it
                                
                                ;Get repcount and store it
                                
002017' 603 16 0 00 000004              TLNE    CURSTS,(ONETIM)         ;First time here?
002020' 254 00 0 00 002024'             JRST    ACT06A                  ;No, go around
002021' 260 17 0 00 003364'             GO      GETREP                  ;Get repcount and
002022' 200 02 0 00 000276'             MOVE    AC2,CUREXE              ; Store
002023' 5C3 00 0 02 000001              HRLM    AC0,RPCNT(AC2)          ; It
                                
                                ;Get the othernode field and store it.
                                
002024' 201 01 0 03 000002      ACT06A: MOVEI   AC1,2(AC3)              ;Get "OTHERNODE" address word
002025' 135 00 0 00 000331'             LDB     AC0,OTHE                ;Get othernode
002026' 202 00 0 00 000232'             MOVEM   AC0,OTHERN              ;Store it
                                
                                ;Set up the status byte for success.
                                
002027' 201 00 0 00 000000              MOVEI   AC0,SUCCES              ;Load successful status
002030' 260 17 0 00 003450'             GO      LDSTUS                  ;Do it
                                
                                ;See if we have a connection open to this other node
                                
002031' 476 00 0 00 000002      ACT06B: SETOM   AC2                     ;Init index pointer
002032' 201 03 0 00 000040              MOVEI   AC3,CBLSTE              ;Get size of cblst
002033' 350 00 0 00 000002      ACT06C: AOS     AC2                     ;Increment index
002034' 200 04 0 02 010556'             MOVE    AC4,CBLST(AC2)          ;Get cblk address
002035' 336 00 0 04 000000              SKIPN   CID(AC4)                ;Is this cblk being used?
002036' 254 00 0 00 002045'             JRST    ACT06D                  ;No, look for another
002037' 550 01 0 04 000005              HRRZ    AC1,LOCPRT(AC4)         ;Get the controllers ci number
002040' 312 01 0 00 000256'             CAME    AC1,CIPORT              ;Is this 3rd party on the same ci?
002041' 254 00 0 00 002045'             JRST    ACT06D                  ;No
002042' 550 01 0 04 000004              HRRZ    AC1,NODE(AC4)           ;Get this cblks 3rd party node #
002043' 316 01 0 00 000232'             CAMN    AC1,OTHERN              ;Do we have a connection open?
002044' 254 00 0 00 002050'             JRST    ACT06E                  ;Yes, continue
002045' 367 03 0 00 002033'     ACT06D: SOJG    AC3,ACT06C              ;No, keep looking
002046' 201 00 0 00 000366              MOVEI   AC0,NOCONN              ;Yes, do no connection exist
002047' 254 00 0 00 002102'             JRST    ACT06X                  ;Exit, take +1 error rtn
```

```
                                        ;Now do the "RESET"

002050' 201 01 0 00 000105      ACT06E: MOVEI   AC1,.DGRST             ;Store the "RESET"
002051' 202 01 0 00 006342'             MOVEM   AC1,DIAGAG              ; Function code
002052' 514 01 0 00 000256'             HRLZ    AC1,CIPORT              ;Port # to lh
002053' 540 01 0 00 000232'             HRR     AC1,OTHERN              ;Node # to rh
002054' 202 01 0 00 006343'             MOVEM   AC1,DIAGAG+1            ;Put the port#,,node# to arg block

                                        ;Look at the extend byte

002055' 200 01 0 00 000276'             MOVE    AC1,CUREXE              ;Get rhblk
002056' 200 02 0 01 000003              MOVE    AC2,CTPPKT(AC1)         ;Now get the rblk
002057' 201 01 0 02 000002              MOVEI   AC1,2(AC2)              ;Get extend address word
002060' 135 00 0 00 000322'             LDB     AC0,EXTE                ;Get extend
002061' 476 00 0 00 006344'             SETOM   DIAGAG+2                ;Set up for "UNCONDITIONAL RESET"
002062' 302 00 0 00 000000              CAIE    AC0,0                   ;Do an "UNCONDITIONAL RESET"?
002063' 402 00 0 00 006344'             SETZM   DIAGAG+2                ;No

002064' 200 01 0 00 261146              MOVE    AC1,[-3,,DIAGAG]        ;Neg length of arg block,,address
002065' 260 17 0 00 006247'             GO      DIAGJ                   ;Do the "DIAG" jsys
002066' 254 00 0 00 002070'             JRST    ACT06F                  ;Error rtn
002067' 254 00 0 00 002072'             JRST    ACT06H                  ;Success

002070' 201 00 0 00 000373      ACT06F: MOVEI   AC0,FUNCFL              ;Do function failure
002071' 254 00 0 00 002102'             JRST    ACT06X                  ;Do +1 error rtn

                                        ;Increment the actcount field

002072' 603 16 0 00 000004      ACT06H: TLNE    CURSTS,(ONETIM)         ;First time here?
002073' 254 00 0 00 002076'             JRST    ACT06J                  ;No, go around
002074' 476 00 0 00 000000              SETOM   AC0                     ;Init actcount to -1
002075' 260 17 0 00 003402'             GO      STRACT                  ; For the first time
002076' 260 17 0 00 003364'     ACT06J: GO      GETACT                  ;Get actcount
002077' 350 00 0 00 000000              AOS     AC0                     ;Increment it
002100' 260 17 0 00 003402'             GO      STRACT                  ;Now store it
002101' 254 00 0 00 002104'             JRST    ACT06S                  ;Take the +2 good rtn

002102' 260 17 0 00 003450'     ACT06X: GO      LDSTUS                  ;Load the status byte
002103' 334 00 0 00 000000              SKIPA
002104' 350 00 0 17 777772      ACT06S: AOS     -6(P)                   ;Normal skip return
002105' 262 17 0 00 000005              GET     AC5                     ;Restore ac's
002106' 262 17 0 00 000004              GET     AC4
002107' 262 17 0 00 000003              GET     AC3
002110' 262 17 0 00 000002              GET     AC2
002111' 262 17 0 00 000001              GET     AC1
002112' 262 17 0 00 000000              GET     AC0
002113' 263 17 0 00 000000              RTN                             ;Error return
```

```
                        SUBTTL  CTP Generate Start

                ;This routine will generate a start packet on the the ci.  Before we get here
                ;we should have received a "CONNECT" request from the cint, establishing a 3rd
                ;party connection to a responder.  This routine will only be used with a cint.

002114' 332 00 0 00 000216'     GENSTR: TRMSG   (Opcode: Generate start)
002115' 260 17 0 00 261154
002116' 261 17 0 00 000000              PUT     AC0                     ;Save ac's
002117' 261 17 0 00 000001              PUT     AC1
002120' 261 17 0 00 000002              PUT     AC2
002121' 603 16 0 00 000004              TLNE    CURSTS,(ONETIM)         ;First time here?
002122' 254 00 0 00 002131'             JRST    GENST1                  ;No, go around the delay
002123' 260 17 0 00 003543'             GO      CHGOPC                  ;Change opcode from a req to a res.
002124' 260 17 0 00 002154'             GO      VAL07                   ;Check the validity of the ctp request
002125' 254 00 0 00 002145'             JRST    GENST2                  ;Error in the validity check
002126' 260 17 0 00 004170'             GO      EXDLY                   ;Do the delay
002127' 254 00 0 00 002147'             JRST    GENST3                  ;+1  Delay has been started, go around
002130' 254 00 0 00 002145'             JRST    GENST2                  ;+2  "CFORK" Failure
002131' 260 17 0 00 002203'     GENST1: GO      ACT07                   ;Perform action for the request
002132' 254 00 0 00 002145'             JRST    GENST2                  ;Error return
002133' 200 02 0 00 000276'             MOVE    AC2,CUREXE              ;Get rblk address
002134' 554 01 0 02 000001              HLRZ    AC1,RPCNT(AC2)          ;Offset to the repcount
002135' 370 00 0 00 000001              SOS     AC1                     ;We start counting at 0 so subtract rep.
002136' 260 17 0 00 003364'             GO      GETACT                  ;Get the actcount
002137' 312 00 0 00 000001              CAME    AC0,AC1                 ;Actcount = repcount
002140' 254 00 0 00 002147'             JRST    GENST3                  ;No
002141' 661 16 0 00 000002              TLO     CURSTS,(DONE)           ;Set the done bit
002142' 621 16 0 00 000001              TLZ     CURSTS,(GOBIT)          ;Clear the gobit
002143' 200 01 0 00 000276'             MOVE    AC1,CUREXE              ;Get the rhblk
002144' 202 16 0 01 000002              MOVEM   CURSTS,STATUS(AC1)      ;Store the status
002145' 260 17 0 00 004124'     GENST2: GO      SEND                    ;Send the ctp response message
002146' 255 00 0 00 000000              JFCL                            ;Virtual circuit gone return
002147' 661 16 0 00 000004      GENST3: TLO     CURSTS,(ONETIM)         ;Set the onetim flag
002150' 262 17 0 00 000002              GET     AC2
002151' 262 17 0 00 000001              GET     AC1
002152' 262 17 0 00 000000              GET     AC0
002153' 263 17 0 00 000000              RTN                             ;Return
```

```
                                        ;Check the validity of the ctp request

002154' 261 17 0 00 000000      VAL07:  PUT     AC0                     ;Save ac's
002155' 261 17 0 00 000001              PUT     AC1
002156' 261 17 0 00 000002              PUT     AC2
002157' 200 01 0 00 000276'             MOVE    AC1,CUREXE              ;Get rhblk
002160' 200 02 0 01 000003              MOVE    AC2,CTPPKT(AC1)         ;Now get the rblk
002161' 201 01 0 02 000002              MOVEI   AC1,2(AC2)              ;Get extend address word
002162' 135 00 0 00 000322'             LDB     AC0,EXTE                ;Get extend
002163' 307 00 0 00 000001              CAIG    AC0,1                   ;Value bigger than 1?
002164' 254 00 0 00 002167'             JRST    VAL07A                  ;No, continue
002165' 201 00 0 00 000376              MOVEI   AC0,NONSEN              ;Yes, do request not understood
002166' 254 00 0 00 002174'             JRST    VAL07X                  ;Take the +1 error return

                                        ;Validate the "OTHERNODE" byte

002167' 201 01 0 03 000002      VAL07A: MOVEI   AC1,2(AC3)              ;Get "OTHERNODE" address word
002170' 135 00 0 00 000331'             LDB     AC0,OTHE                ;Get othernode
002171' 331 00 0 00 000000              SKIPL   AC0                     ;Value a negative number?
002172' 254 00 0 00 002176'             JRST    VAL07S                  ;No, continue
002173' 201 00 0 00 000376              MOVEI   AC0,NONSEN              ;Yes, do request not understood

002174' 260 17 0 00 003450'     VAL07X: GO      LDSTUS                  ;Load the status byte in the rblk
002175' 334 00 0 00 000000              SKIPA                           ;Take the +1 error return
002176' 350 00 0 17 777775      VAL07S: AOS     -3(P)                   ;+2 Return
002177' 262 17 0 00 000002              GET     AC2
002200' 262 17 0 00 000001              GET     AC1
002201' 262 17 0 00 000000              GET     AC0
002202' 263 17 0 00 000000              RTN                             ;Return
```

```
                                ;Do the function the ctp request packet demanded

                                ACT07:  PUT     AC0                     ;Save ac's
002203' 261 17 0 00 000000              PUT     AC1
002204' 261 17 0 00 000001              PUT     AC2
002205' 261 17 0 00 000002              PUT     AC3
002206' 261 17 0 00 000003              PUT     AC4
002207' 261 17 0 00 000004              PUT     AC5
002210' 261 17 0 00 000005

                                ;Get the controllers ci number and store it.

002211' 550 01 0 00 000016              HRRZ    AC1,CURSTS              ;Get the cblk in ac1
002212' 200 01 0 01 000005              MOVE    AC1,LOCPRT(AC1)         ;Now get the local port number(ci #)
002213' 202 01 0 00 000256'             MOVEM   AC1,CIPORT              ; And store it

                                ;Get repcount and store it

002214' 603 16 0 00 000004              TLNE    CURSTS,(ONETIM)         ;First time here?
002215' 254 00 0 00 002221'             JRST    ACT07A                  ;No, go around
002216' 260 17 0 00 003364'             GO      GETREP                  ;Get repcount and
002217' 200 02 0 00 000276'             MOVE    AC2,CUREXE              ; Store
002220' 506 00 0 02 000001              HRLM    AC0,RPCNT(AC2)          ; It

                                ;Get the ouhernode field and store it.

002221' 201 01 0 03 000002      ACT07A: MOVEI   AC1,2(AC3)              ;Get "OTHERNODE" address word
002222' 135 00 0 00 000331'             LDB     AC0,OTHE                ;Get othernode
002223' 202 00 0 00 000232'             MOVEM   AC0,OTHERN              ;Store it

                                ;Get the start address and store it.

002224' 201 01 0 03 000003              MOVEI   AC1,3(AC3)              ;Get "START ADDRESS" address word
002225' 135 00 0 00 000342'             LDB     AC0,STAR                ;Get the start address
002226' 202 00 0 00 000252'             MOVEM   AC0,STRTAD              ;Store it

                                ;Set up the status byte for success.

002227' 201 00 0 00 000000              MOVEI   AC0,SUCCES              ;Load successful status
002230' 260 17 0 00 003450'             GO      LDSTUS                  ;Do it
```

```
1700
1701                                    ;See if we have a connection open to this other node
1702
1703  002231' 476 00 0 00 000002  ACT07B: SETOM   AC2                    ;Init index pointer
1704  002232' 201 03 0 00 000040         MOVEI   AC3,CBLSTE              ;Get size of cblst
1705  002233' 350 00 0 00 000002  ACT07C: AOS     AC2                    ;Increment index
1706  002234' 200 04 0 02 010556'        MOVE    AC4,CBLST(AC2)          ;Get cblk address
1707  002235' 336 00 0 04 000000         SKIPN   CID(AC4)                ;Is this cblk being used?
1708  002236' 254 00 0 00 002245'        JRST    ACT07D                  ;No, look for another
1709  002237' 550 01 0 04 000005         HRRZ    AC1,LOCPRT(AC4)         ;Get the controllers ci number
1710  002240' 312 01 0 00 000256'        CAME    AC1,CIPORT              ;Is this 3rd party on the same ci?
1711  002241' 254 00 0 00 002245'        JRST    ACT07D                  ;No
1712  002242' 550 01 0 04 000004         HRRZ    AC1,NODE(AC4)           ;Get this cblks 3rd party node #
1713  002243' 316 01 0 00 000232'        CAMN    AC1,OTHERN              ;Do we have a connection open?
1714  002244' 254 00 0 00 002250'        JRST    ACT07E                  ;Yes, continue
1715  002245' 367 03 0 00 002233' ACT07D: SOJG    AC3,ACT07C              ;No, keep looking
1716  002246' 201 00 0 00 000366         MOVEI   AC0,NOCONN              ;Yes, do no connection exist
1717  002247' 254 00 0 00 002306'        JRST    ACT07X                  ;Exit, take +1 error rtn
1718
1719                                    ;Now do the start
1720
1721  002250' 201 01 0 00 000106  ACT07E: MOVEI   AC1,..DGSTR             ;Store the "START"
1722  002251' 202 01 0 00 006342'        MOVEM   AC1,DIAGAG              ; Function code
1723  002252' 514 01 0 00 000256'        HRLZ    AC1,CIPORT              ;Port # to lh
1724  002253' 540 01 0 00 000232'        HRR     AC1,OTHERN              ;Node # to rh
1725  002254' 202 01 0 00 006343'        MOVEM   AC1,DIAGAG+1            ;Put the port#,,node# to arg block
1726
1727                                    ;Look at the extend byte
1728
1729  002255' 200 01 0 00 000276'        MOVE    AC1,CUREXE              ;Get rhblk
1730  002256' 200 02 0 01 000003         MOVE    AC2,CTPPKT(AC1)         ;Now get the rblk
1731  002257' 201 01 0 02 000002         MOVEI   AC1,2(AC2)              ;Get extend address word
1732  002260' 135 00 0 00 000322'        LDB     AC0,EXTE                ;Get extend
1733  002261' 402 00 0 00 006344'        SETZM   DIAGAG+2                ;Set up to use "DEFAULT ADDRESS"
1734  002262' 302 00 0 00 000001         CAIE    AC0,1                   ;Use the "DEFAULT ADDRESS"?
1735  002263' 254 00 0 00 002270'        JRST    ACT07G                  ;Yes
```

```
1736
1737                                        ;Use the supplied start address
1738
1739  002264' 200 00 0 00 000252'           MOVE    AC0,STRTAD          ;Get the start address
1740  002265' 242 00 0 00 777774            LSH     AC0,-4              ;Shift right 4 places
1741  002266' 260 17 0 00 003463'           GO      P11T10              ;Get it in pdp10 format
1742  002267' 202 00 0 00 006344'           MOVEM   AC0,DIAGAG+2        ;Put it in the argument block
1743
1744  002270' 200 01 0 00 261146  ACT07G: MOVE    AC1,[-3,,DIAGAG]    ;Neg length of arg block,,address
1745  002271' 260 17 0 00 006247'           GO      DIAGJ               ;Do the "DIAG" jsys
1746  002272' 254 00 0 00 002274'           JRST    ACT07F              ;Error rtn
1747  002273' 254 00 0 00 002276'           JRST    ACT07H              ;Success
1748
1749  002274' 201 00 0 00 000373  ACT07F: MOVEI   AC0,FUNCFL           ;Do function failure
1750  002275' 254 00 0 00 002306'           JRST    ACT07X              ;Do +1 error rtn
1751
1752                                        ;Increment the actcount field
1753
1754  002276' 603 16 0 00 000004  ACT07H: TLNE    CURSTS,(ONETIM)     ;First time here?
1755  002277' 254 00 0 00 002302'           JRST    ACT07J              ;No, go around
1756  002300' 476 00 0 00 000000            SETOM   AC0                 ;Init actcount to -1
1757  002301' 260 17 0 00 003402'           GO      STRACT              ; For the first time
1758  002302' 260 17 0 00 003364' ACT07J: GO      GETACT              ;Get actcount
1759  002303' 350 00 0 00 000000            AOS     AC0                 ;Increment it
1760  002304' 260 17 0 00 003402'           GO      STRACT              ;Now store it
1761  002305' 254 00 0 00 002310'           JRST    ACT07S              ;Take the +2 good rtn
1762
1763  002306' 260 17 0 00 003450' ACT07X: GO      LDSTUS              ;Load the status byte
1764  002307' 334 00 0 00 000000            SKIPA
1765  002310' 350 00 0 17 777772  ACT07S: AOS     -6(P)               ;Normal skip return
1766  002311' 262 17 0 00 000005            GET     AC5                 ;Restore ac's
1767  002312' 262 17 0 00 000004            GET     AC4
1768  002313' 262 17 0 00 000003            GET     AC3
1769  002314' 262 17 0 00 000002            GET     AC2
1770  002315' 262 17 0 00 000001            GET     AC1
1771  002316' 262 17 0 00 000000            GET     AC0
1772  002317' 263 17 0 00 000000            RTN                         ;Error return
```

```
                                        SUBTTL  CTP Unsolicited Activity

                                ;This routine will either start or stop the monitor from executing
                                ;"REQUEST ID'S"(Polling). This routine will only be used with a cint.

002320' 332 00 0 00 000216'     SSUNSO: TRMSG   (Opcode: Unsolicited activity)
002321' 260 17 0 00 261170
002322' 260 17 0 00 003543'             GO      CHGOPC                  ;Change opcode from a req to a res.
002323' 260 17 0 00 002335'             GO      VAL08                   ;Check validity of the ctp request
002324' 260 17 0 00 002342'             GO      ACT08                   ;Perform action for the request
002325' 255 00 0 00 000000              JFCL                            ;Error return
002326' 661 16 0 00 000002              TLO     CURSTS,(DONE)           ;Set the done bit
002327' 621 16 0 00 000001              TLZ     CURSTS,(GOBIT)          ;Clear the gobit
002330' 200 01 0 00 000276'             MOVE    AC1,CUREXE              ;Get the rhblk
002331' 202 16 0 01 000002              MOVEM   CURSTS,STATUS(AC1)      ;Store the status
002332' 260 17 0 00 004124'     SSUNS1: GO      SEND                    ;Send back the ctp response
002333' 255 00 0 00 000000              JFCL                            ;Error return
002334' 263 17 0 00 000000              RTN                             ;Return

                                ;Check the validity of the ctp request

002335' 261 17 0 00 000000      VAL08:  PUT     AC0                     ;Save ac's
002336' 201 00 0 00 000000              MOVEI   AC0,SUCCES              ;Zero the
002337' 260 17 0 00 003450'             GO      LDSTUS                  ; Status byte in the ctp packet
002340' 262 17 0 00 000000              GET     AC0                     ;Restore ac's
002341' 263 17 0 00 000000              RTN                             ;Return
```

```
1800
1801                                    ;Do the function the ctp request packet demanded
1802
1803  002342' 261 17 0 00 000000  ACT08:  PUT     AC0                     ;Save ac's
1804  002343' 261 17 0 00 000001          PUT     AC1
1805  002344' 261 17 0 00 000002          PUT     AC2
1806  002345' 261 17 0 00 000003          PUT     AC3
1807  002346' 261 17 0 00 000004          PUT     AC4
1808  002347' 200 00 0 00 000256'         MOVE    AC0,CIPORT              ;Get port number
1809  002350' 202 00 0 00 006343'         MOVEM   AC0,DIAGAG+1            ;Store in arg block
1810  002351' 200 02 0 00 000276'         MOVE    AC2,CUREXE              ;Get the rhblk
1811  002352' 200 01 0 02 000003          MOVE    AC1,CTPPKT(AC2)         ;Get the rblk address
1812  002353' 201 01 0 01 000002          MOVEI   AC1,2(AC1)              ;Get unsolicited activity address word
1813  002354' 135 00 0 00 000332'         LDB     AC0,NOAC                ;Get unsol act field
1814  002355' 202 00 0 00 000253'         MOVEM   AC0,UNSACT              ;Store it
1815  002356' 336 00 0 00 000000          SKIPN   AC0                     ;0 = Start polling, 1 = stop polling
1816  002357' 254 00 0 00 002367'         JRST    ACT08C                  ;Start the poller
1817
1818                                    ;Stop the poller
1819
1820  002360' 201 01 0 00 000111          MOVEI   AC1,.DGENB              ;Store the 'POLLER'
1821  002361' 202 01 0 00 006342'         MOVEM   AC1,DIAGAG              ; Function code
1822  002362' 402 00 0 00 006344'         SETZM   DIAGAG+2                ;Stop the poller
1823  002363' 200 01 0 0C ~~1146          MOVE    AC1,[-3,,DIAGAG]        ;Neg length of arg block,,address
1824  002364' 260 17 0 00 uu6247'         GO      DIAGJ                   ;Do the 'DIAG' jsys
1825  002365' 254 00 0 00 002376'          JRST    ACT08A                  ;Error return
1826  002366' 254 00 0 00 002401'         JRST    ACT08S                  ;Success
1827
1828                                    ;Start the poller
1829
1830  002367' 201 01 0 00 000111  ACT08C: MOVEI   AC1,.DGENB              ;Store the 'POLLER'
1831  002370' 202 01 0 00 006342'         MOVEM   AC1,DIAGAG              ; Function code
1832  002371' 350 00 0 00 006344'         AOS     DIAGAG+2                ;Start the poller
1833  002372' 200 01 0 00 261146          MOVE    AC1,[-3,,DIAGAG]        ;Neg length of arg block,,address
1834  002373' 260 17 0 00 006247'         GO      DIAGJ                   ;Do the 'DIAG' jsys
1835  002374' 254 00 0 00 002376'          JRST    ACT08A                  ;Error return
1836  002375' 254 00 0 00 002401'         JRST    ACT08S                  ;Success
1837
1838  002376' 201 00 0 00 000373  ACT08A: MOVEI   AC0,FUNCFL              ;Do function failure
1839  002377' 260 17 0 00 003450' ACT08X: GO      LDSTUS                  ;Load the status byte
1840  002400' 334 00 0 00 000000          SKIPA
1841  002401' 350 00 0 17 777773  ACT08S: AOS     -5(P)                   ;Normal skip return
1842  002402' 262 17 0 00 000004          GET     AC4
1843  002403' 262 17 0 00 000003          GET     AC3
1844  002404' 262 17 0 00 000002          GET     AC2
1845  002405' 262 17 0 00 000001          GET     AC1
1846  002406' 262 17 0 00 000000          GET     AC0
1847  002407' 263 17 0 00 000000          RTN
```

```
                                        SUBTTL  CTP Configuration

                                ;This routine returns information(path status) about another system back to the
                                ;controller.

002410' 332 00 0 00 000216'     CNFDAT: TRMSG   (Opcode: Configuration)
002411' 260 17 0 00 261202
002412' 260 17 0 00 003543'             GO      CHGOPC                  ;Change opcode from a req to a res.
002413' 260 17 0 00 002425'             GO      VAL09                   ;Check the validity of the ctp request
002414' 254 00 0 00 002422'             JRST    CNFDA1                  ;Error in the validity check
002415' 260 17 0 00 002447'             GO      ACT09                   ;Perform action for the request
002416' 661 16 0 00 000002              TLO     CURSTS,(DONE)           ;Set the done bit
002417' 621 16 0 00 000001              TLZ     CURSTS,(GOBIT)          ;Clear the gobit
002420' 200 01 0 00 000276'             MOVE    AC1,CUREXE              ;Get the rhblk
002421' 202 16 0 01 000002              MOVEM   CURSTS,STATUS(AC1)      ;Store the status
002422' 260 17 0 00 004124'     CNFDA1: GO      SEND                    ;Send the ctp response message
002423' 255 00 0 00 000000              JFCL                            ;Virtual circuit gone return
002424' 263 17 0 00 000000              RTN                             ;Return

                                ;Check the validity of the ctp request

002425' 261 17 0 00 000000      VAL09:  PUT     AC0                     ;Save ac's
002426' 261 17 0 00 000001              PUT     AC1
002427' 261 17 0 00 000002              PUT     AC2
002430' 200 01 0 00 000276'             MOVE    AC1,CUREXE              ;Get rhblk
002431' 200 02 0 01 000003              MOVE    AC2,CTPPKT(AC1)         ;Now get the rblk
002432' 201 01 0 02 000002              MOVE    AC1,2(AC2)              ;Get othernode address word
002433' 135 00 0 00 000331'             LDB     AC0,OTHE                ;Get othernode field
002434' 202 00 0 00 000232'             MOVEM   AC0,OTHERN              ;Store it
002435' 331 00 0 00 000000              SKIPL   AC0                     ;Othernode a negative number?
002436' 255 00 0 00 002442'             JRST    VAL09S                  ;No, continue
002437' 201 00 0 00 000376              MOVEI   AC0,NONSEN              ;Yes, do request not understood

002440' 260 17 0 00 003450'     VAL09X: GO      LDSTUS                  ;Load the status byte in the rblk
002441' 334 00 0 00 000000              SKIPA                           ;Take +1 error return
002442' 350 00 0 17 777775      VAL09S: AOS     -3(P)                   ;+2 Return
002443' 262 17 0 00 000002              GET     AC2
002444' 262 17 0 00 000001              GET     AC1
002445' 262 17 0 00 000000              GET     AC0
002446' 263 17 0 00 000000              RTN                             ;Return
```

```
                                ;Do the function the ctp request packet demanded

                                ACT09:  PUT     AC0                     ;Save ac's
002447' 261 17 0 00 000000              PUT     AC1
002450' 261 17 0 00 000001              PUT     AC2
002451' 261 17 0 00 000002              PUT     AC3
002452' 261 17 0 00 000003              PUT     AC4
002453' 261 17 0 00 000004

                                ;Set up the status byte for success.

002454' 201 00 0 00 000000              MOVEI   AC0,SUCCES              ;Load successful status
002455' 260 17 0 00 003450'             GO      LDSTUS                  ;Do it

002456' 402 00 0 00 000001              SETZM   AC1                     ;Don't use cid
002457' 200 02 0 00 000232'             MOVE    AC2,OTHERN              ;Use node number
002460' 260 17 0 00 005737'             GO      CONFIG                  ;Get information about this node
002461' 254 00 0 00 002505'              JRST   ACT09A                  ;Node does not exists
002462' 554 03 0 00 005772'             HLRZ    AC3,CNFARG+3            ;Get "VIRTUAL CIRCUIT STATE" field
002463' 200 01 0 00 000276'             MOVE    AC1,CUREXE              ;Get rhblk
002464' 200 04 0 01 000003              MOVE    AC4,CTPPKT(AC1)         ;Get rblk

                                ;Get path status for this node
002465' 200 00 0 00 000232'             MOVE    AC0,OTHERN              ;Get the node number
002466' 260 17 0 00 003562'             GO      GPRS                    ;Go get path status. Ac0 has status
002467' 254 00 0 00 002505'              JRST   ACT09A                  ;Node doesn't exist

                                ;Get the path 0 status word and store it in the ctp packet.

002470' 402 00 0 00 000003              SETZM   AC3                     ;Assume path status is bad
002471' 554 02 0 00 000000              HLRZ    AC2,AC0                 ;Get path status 0
002472' 332 00 0 00 000002              SKIPE   AC2                     ;Skip if path status is bad
002473' 201 03 0 00 000001              MOVEI   AC3,1                   ;Mark path as good

002474' 201 01 0 04 000002              MOVEI   AC1,2(AC4)              ;Get path 0 status address word
002475' 137 03 0 00 000335'             DPB     AC3,POST                ;Store it in the ctp packet

                                ;Get the path 1 status word and store it in the ctp packet.

002476' 402 00 0 00 000003              SETZM   AC3                     ;Assume path status is bad
002477' 550 02 0 00 000000              HRRZ    AC2,AC0                 ;Get path status 1
002500' 332 00 0 00 000002              SKIPE   AC2                     ;Skip if path status is bad
002501' 201 03 0 00 000001              MOVEI   AC3,1                   ;Mark path as good

002502' 201 01 0 04 000003              MOVEI   AC1,3(AC4)              ;Get path 1 status address word
002503' 137 03 0 00 000337'             DPB     AC3,P1ST                ;Store it in the ctp packet
002504' 254 00 0 00 002507'             JRST    ACT09S                  ;All done
002505' 201 00 0 00 000370      ACT09A: MOVEI   AC0,NODEUN              ;Yes, do node unknown to sca
002506' 260 17 0 00 003450'             GO      LDSTUS                  ;Load the status byte in the rblk
002507' 262 17 0 00 000004      ACT09S: GET     AC4                     ;Restore ac's
002510' 262 17 0 00 000003              GET     AC3
002511' 262 17 0 00 000002              GET     AC2
002512' 262 17 0 00 000001              GET     AC1
002513' 262 17 0 00 000000              GET     AC0
002514' 263 17 0 00 000000              RTN                             ;Error return
```

```
                                        SUBTTL  CTP Counter Read

                                ;This routine reads the counters in our port and returns them to the
                                ;controller.

002515' 332 00 0 00 000216'     CNTRRD: TRMSG   (Opcode: Counter read)
002516' 260 17 0 00 261214
002517' 260 17 0 00 003543'             GO      CHGOPC                  ;Change opcode from a req to a res.
002520' 260 17 0 00 002532'             GO      VAL10                   ;Check the validity of the ctp request
002521' 254 00 0 00 002523'             JRST    CNTRR1                  ;Error in the validity check
002522' 260 17 0 00 002554'             GO      ACT10                   ;Perform action for the request
002523' 260 17 0 00 004124'     CNTRR1: GO      SEND                    ;Send the ctp response message
002524' 255 00 0 00 000000              JFCL                            ;Virtual circuit gone return
002525' 661 16 0 00 000002              TLO     CURSTS,(DONE)           ;Set the done bit
002526' 621 16 0 00 000001              TLZ     CURSTS,(GOBIT)          ;Clear the gobit
002527' 200 01 0 00 000276'             MOVE    AC1,CUREXE              ;Get the rhblk
002530' 202 16 0 01 000002              MOVEM   CURSTS,STATUS(AC1)      ;Store the status
002531' 263 17 0 00 000000              RTN                             ;Return

                                ;Check the validity of the ctp request

002532' 261 17 0 00 000000      VAL10:  PUT     AC0                     ;Save ac's
002533' 261 17 0 00 000001              PUT     AC1
002534' 261 17 0 00 000002              PUT     AC2
002535' 200 01 0 00 000276'             MOVE    AC1,CUREXE              ;Get rhblk
002536' 200 02 0 01 000003              MOVE    AC2,CTPPKT(AC1)         ;Now get the rblk
002537' 201 01 0 02 000002              MOVEI   AC1,2(AC2)              ;Get othernode address word
002540' 135 00 0 00 000331'             LDB     AC0,OTHE                ;Get othernode field
002541' 202 00 0 00 000232'             MOVEM   AC0,OTHERN              ;Store it
002542' 331 00 0 00 000000              SKIPL   AC0                     ;Value a negative number?
002543' 254 00 0 00 002547'             JRST    VAL10S                  ;No, continue
002544' 201 00 0 00 000376              MOVEI   AC0,NONSEN              ;Yes, do request not understood

002545' 260 17 0 00 003450'     VAL10X: GO      LDSTUS                  ;Load the status byte in the rblk
002546' 334 00 0 00 000000              SKIPA                           ;Take +1 error return
002547' 350 00 0 17 777775      VAL10S: AOS     -3(P)                   ;+2 Return
002550' 262 17 0 00 000002              GET     AC2
002551' 262 17 0 00 000001              GET     AC1
002552' 262 17 0 00 000000              GET     AC0
002553' 263 17 0 00 000000              RTN                             ;Return
```

```
                                        ;Do the function the ctp request packet demanded

002554' 261 17 0 00 000000      ACT10:  PUT     AC0                     ;Save ac's
002555' 261 17 0 00 000001              PUT     AC1
002556' 261 17 0 00 000002              PUT     AC2

                                        ;Set up the status byte for success.

002557' 201 00 0 00 000000              MOVEI   AC0,SUCCES              ;Load successful status
002560' 260 17 0 00 003450'             GO      LDSTUS                  ;Do it

                                        ;Get the controllers ci number and store it.

002561' 550 01 0 00 000016              HRRZ    AC1,CURSTS              ;Get the cblk in ac1
002562' 200 01 0 01 000005              MOVE    AC1,LOCPRT(AC1)         ;Now get the local port number(ci #)
002563' 202 01 0 00 000256'             MOVEM   AC1,CIPORT              ; And store it

                                        ;Read the counters

002564' 201 01 0 00 000107              MOVEI   AC1,.DGCTR              ;Get the "PORT COUNTER" function
002565' 202 01 0 00 006342'             MOVEM   AC1,DIAGAG              ;Store it in argument block
002566' 515 01 0 00 000007              HRLZI   AC1,7                   ;Get port number
002567' 541 01 0 00 000003              HRRI    AC1,3                   ;Read the counters function code
002570' 202 01 0 00 006343'             MOVEM   AC1,DIAGAG+1            ;Store it in argument block
002571' 402 00 0 00 006344'             SETZM   DIAGAG+2                ;Zero rest of the
002572' 402 00 0 00 006345'             SETZM   DIAGAG+3                ; Arg block

002573' 200 01 0 00 261221              MOVE    AC1,[-^D21,,DIAGAG]     ;Neg length of arg block,,address
002574' 260 17 0 00 006247'             GO      DIAGJ                   ;Do the "DIAG" jsys
002575' 254 00 0 00 002706'              JRST   ACT10B                  ;Error return

                                ;Put the data in the packet we will ship back

002576' 200 01 0 00 000276'             MOVE    AC1,CUREXE              ;Get rhblk
002577' 200 02 0 01 000003              MOVE    AC2,CTPPKT(AC1)         ;Get rblk address
002600' 271 02 0 00 000003              ADDI    AC2,3                   ;+3 To get to path a ack count storage

002601' 200 00 0 00 006346'             MOVE    AC0,DIAGAG+4            ;Get the path 0 ack count
002602' 260 17 0 00 003502'             GO      P10T11                  ;Change it to pdp11 word
002603' 242 00 0 00 000004              LSH     AC0,4                   ;Left justify it
002604' 202 00 1 00 000002              MOVEM   AC0,@AC2                ;Store it in the ctp packet
```

```
2024
2025  002605' 200 00 0 00 006347'          MOVE    AC0,DIAGAG+5          ;Get the path 0 nak count
2026  002606' 260 17 0 00 003502'          GO      P10T11                ;Change it to pdp11 word
2027  002607' 350 00 0 00 000002           AOS     AC2                   ;Increment rblk address
2028  002610' 242 00 0 00 000004           LSH     AC0,4                 ;Left justify it
2029  002611' 202 00 1 00 000002           MOVEM   AC0,@AC2              ;Store it in the ctp packet
2030
2031  002612' 200 00 0 00 006350'          MOVE    AC0,DIAGAG+6          ;Get the path 0 no response count
2032  002613' 260 17 0 00 003502'          GO      P10T11                ;Change it to pdp11 word
2033  002614' 350 00 0 00 000002           AOS     AC2                   ;Increment rblk address
2034  002615' 242 00 0 00 000004           LSH     AC0,4                 ;Left justify it
2035  002616' 202 00 1 00 000002           MOVEM   AC0,@AC2              ;Store it in the ctp packet
2036
2037  002617' 200 00 0 00 006351'          MOVE    AC0,DIAGAG+7          ;Get the path 1 ack count
2038  002620' 260 17 0 00 003502'          GO      P10T11                ;Change it to pdp11 word
2039  002621' 350 00 0 00 000002           AOS     AC2                   ;Increment rblk address
2040  002622' 242 00 0 00 000004           LSH     AC0,4                 ;Left justify it
2041  002623' 202 00 1 00 000002           MOVEM   AC0,@AC2              ;Store it in the ctp packet
2042
2043  002624' 200 00 0 00 006352'          MOVE    AC0,DIAGAG+10         ;Get the path 1 nak count
2044  002625' 260 17 0 00 003502'          GO      P10T11                ;Change it to pdp11 word
2045  002626' 350 00 0 00 000002           AOS     AC2                   ;Increment rblk address
2046  002627' 242 00 0 00 000004           LSH     AC0,4                 ;Left justify it
2047  002630' 202 00 1 00 000002           MOVEM   AC0,@AC2              ;Store it in the ctp packet
2048
2049  002631' 200 00 0 00 006353'          MOVE    AC0,DIAGAG+11         ;Get the path 1 no response count
2050  002632' 260 17 0 00 003502'          GO      P10T11                ;Change it to pdp11 word
2051  002633' 350 00 0 00 000002           AOS     AC2                   ;Increment rblk address
2052  002634' 242 00 0 00 000004           LSH     AC0,4                 ;Left justify it
2053  002635' 202 00 1 00 000002           MOVEM   AC0,@AC2              ;Store it in the ctp packet
2054
2055  002636' 200 00 0 00 006354'          MOVE    AC0,DIAGAG+12         ;Get the number of discarded datagrams
2056  002637' 350 00 0 00 000002           AOS     AC2                   ;Increment rblk address
2057  002640' 242 00 0 00 000004           LSH     AC0,4                 ;Left justify it
2058  002641' 202 00 1 00 000002           MOVEM   AC0,@AC2              ;Store it in the ctp packet
2059
2060                                ;Force the release of the counters.  Someone else may have them.
2061
2062  002642' 201 01 0 00 000107           MOVEI   AC1,.DGCTR            ;Get the "PORT COUNTER" function
2063  002643' 202 01 0 00 006342'          MOVEM   AC1,DIAGAG            ;Store it in argument block
2064  002644' 515 01 0 00 000007           HRLZI   AC1,7                 ;Get port number
2065  002645' 541 01 0 00 000001           HRRI    AC1,1                 ;Release the counters function code
2066  002646' 202 01 0 00 006343'          MOVEM   AC1,DIAGAG+1          ;Store it in argument block
2067  002647' 201 01 0 00 000001           MOVEI   AC1,1                 ;Force release counters function
2068  002650' 202 01 0 00 006344'          MOVEM   AC1,DIAGAG+2          ;Store it in argument block
2069  002651' 402 00 0 00 006345'          SETZM   DIAGAG+3              ;Zero rest of the argument block
2070
2071  002652' 200 01 0 00 261221           MOVE    AC1,[-^D21,,DIAGAG]   ;Neg length of arg block,,address
2072  002653' 260 17 0 00 006247'          GO      DIAGJ                 ;Do the "DIAG" jsys
2073  002654' 254 00 0 00 002706'           JRST   ACT10B                ;Error return
```

```
2074
2075                                    ;Get control of the counters.
2076
2077  002655' 201 01 0 00 000107        MOVEI   AC1,,DGCTR              ;Get the "PORT COUNTER" function
2078  002656' 202 01 0 00 006342'       MOVEM   AC1,DIAGAG              ;Store it in argument block
2079  002657' 515 01 0 00 000007        HRLZI   AC1,7                   ;Get port number
2080  002660' 541 01 0 00 000000        HRRI    AC1,0                   ;Get control of counters function code
2081  002661' 202 01 0 00 006343'       MOVEM   AC1,DIAGAG+1            ;Store in arg block
2082  002662' 402 00 0 00 006344'       SETZM   DIAGAG+2                ;Zero rest of the argument block
2083  002663' 402 00 0 00 006345'       SETZM   DIAGAG+3                ;Zero rest of the argument block
2084
2085  002664' 200 01 0 00 261221        MOVE    AC1,[-^D21,,DIAGAG]     ;Neg length of arg block,,address
2086  002665' 260 17 0 00 006247'       GO      DIAGJ                   ;Do the "DIAG" jsys
2087  002666' 254 00 0 00 002706'        JRST    ACT10B                  ;Error return
2088
2089                                    ;Set up the counters for the requested node number.
2090
2091  002667' 201 01 0 00 000107        MOVEI   AC1,,DGCTR              ;Get the "PORT COUNTER" function
2092  002670' 202 01 0 00 006342'       MOVEM   AC1,DIAGAG              ;Store it in argument block
2093  002671' 515 01 0 00 000007        HRLZI   AC1,7                   ;Get port number
2094  002672' 541 01 0 00 000002        HRRI    AC1,2                   ;Set up counters function code
2095  002673' 202 01 0 00 006343'       MOVEM   AC1,DIAGAG+1            ;Store in arg block
2096  002674' 205 01 0 00 777777        MOVSI   AC1,-1                  ;Load -1,,0 to reset counters function
2097  002675' 202 01 0 00 006344'       MOVEM   AC1,DIAGAG+2            ;Store it in argument block
2098  002676' 200 01 0 00 000232'       MOVE    AC1,OTHERN              ;Get the node number
2099  002677' 202 01 0 00 006345'       MOVEM   AC1,DIAGAG+3            ;Store it in argument block
2100  002700' 200 01 0 00 261221        MOVE    AC1,[-^D21,,DIAGAG]     ;Neg length of arg block,,address
2101  002701' 260 17 0 00 006247'       GO      DIAGJ                   ;Do the "DIAG" jsys
2102  002702' 254 00 0 00 002706'        JRST    ACT10B                  ;Error return
2103  002703' 254 00 0 00 002710'       JRST    ACT10S                  ;All done
2104
2105  002704' 201 00 0 00 000376  ACT10A: MOVEI   AC0,NONSEN             ;Yes, do request not understood
2106  002705' 334 00 0 00 000000        SKIPA
2107  002706' 201 00 0 00 000375  ACT10B: MOVEI   AC0,NORESO             ;Yes, do resource failure
2108
2109  002707' 260 17 0 00 003450' ACT10X: GO      LDSTUS                  ;Load the status byte in the rblk
2110  002710' 262 17 0 00 000002  ACT10S: GET     AC2                     ;Restore ac's
2111  002711' 262 17 0 00 000001         GET     AC1
2112  002712' 262 17 0 00 000000         GET     AC0
2113  002713' 263 17 0 00 000000         RTN                             ;Error return
```

```
                                        SUBTTL  CTP Connect

                                ;This routine starts up a 3rd party connection to a responder provided one does
                                ;not already exist to that same responder.  We come here before we do other ctp
                                ;requests that require 3rd party.

002714' 332 00 0 00 000216'     CTPCON: TRMSG   (Opcode: Connect)
002715' 260 17 0 00 261226
002716' 261 17 0 00 000000              PUT     AC0                     ;Save ac's
002717' 607 16 0 00 000004              TLNN    CURSTS,(ONETIM)         ;Have we been here before?
002720' 254 00 0 00 002733'             JRST    CTPCOB                  ;No
002721' 607 16 0 00 000040              TLNN    CURSTS,(OVRDUE)         ;Waited long enough for an answer?
002722' 254 00 0 00 002726'             JRST    CTPCOA                  ;No, go ahead
002723' 201 00 0 00 000373              MOVEI   AC0,FUNCFL              ;Yes, do function failure
002724' 260 17 0 00 003450'             GO      LDSTUS                  ;Load the status byte in the rblk
002725' 254 00 0 00 002746'             JRST    CTPCOC                  ;Yes, send back "FUNCTION FAILURE"
002726' 607 16 0 00 000200      CTPCOA: TLNN    CURSTS,(CONT)           ;Continue bit set?
002727' 254 00 0 00 002750'             JRST    CTPCOX                  ;No, exit
002730' 201 00 0 00 000000              MOVEI   AC0,SUCCES              ;Success to ctp status
002731' 260 17 0 00 003450'             GO      LDSTUS                  ;Load the status byte in the rblk
002732' 254 00 0 00 002746'             JRST    CTPCOC                  ;Yes, send back "SUCCESFUL STATUS"

002733' 260 17 0 00 003543'     CTPCOB: GO      CHGOPC                  ;Change opcode from a req to a res.
002734' 260 17 0 00 002753'             GO      VAL11                   ;Check the validity of the ctp request
002735' 254 00 0 00 002746'             JRST    CTPCOC                  ;Error in the validity check
002736' 260 17 0 00 003650'             GO      CHKVCS                  ;See if we have a v. C. To this node
002737' 254 00 0 00 002746'             JRST    CTPCOC                  ;Yes we already have a 3rd party connection
002740' 260 17 0 00 003002'             GO      ACT11                   ;Perform action for the request
002741' 254 00 0 00 002750'             JRST    CTPCOX                  ;Don't send resp until we are sure a v.C. Is open.
002742' 661 16 0 00 000002              TLO     CURSTS,(DONE)           ;Set the done bit
002743' 621 16 0 00 000001              TLZ     CURSTS,(GOBIT)          ;Clear the gobit
002744' 200 01 0 00 000276'             MOVE    AC1,CUREXE              ;Get the rhblk
002745' 202 16 0 01 000002              MOVEM   CURSTS,STATUS(AC1)      ;Store the status
002746' 260 17 0 00 004124'     CTPCOC: GO      SEND                    ;Send the ctp response message
002747' 255 00 0 00 000000              JFCL                            ;Virtual circuit gone return
002750' 661 16 0 00 000004      CTPCOX: TLO     CURSTS,(ONETIM)         ;We've been here before flag
002751' 262 17 0 00 000000              GET     AC0
002752' 263 17 0 00 000000              RTN                             ;Return
```

```
                                        ;Check the validity of the ctp request
                                        
002753' 261 17 0 00 000000  VAL11:  PUT     AC0                     ;Save ac's
002754' 261 17 0 00 000001          PUT     AC1
002755' 261 17 0 00 000002          PUT     AC2
002756' 200 01 0 00 000276'         MOVE    AC1,CUREXE              ;Get rhblk
002757' 200 02 0 01 000003          MOVE    AC2,CTPPKT(AC1)         ;Now get the rblk
002760' 201 01 0 02 000002          MOVEI   AC1,2(AC2)              ;Get othernode address word
002761' 135 00 0 00 000331'         LDB     AC0,OTHE                ;Get othernode field
002762' 202 00 0 00 000232'         MOVEM   AC0,OTHERN              ;Store it
002763' 331 00 0 00 000000          SKIPL   AC0                     ;Value a negative number?
002764' 254 00 0 00 002767'         JRST    VAL11A                  ;No, continue
002765' 201 00 0 00 000376          MOVEI   AC0,NONSEN              ;Yes, do request not understood
002766' 254 00 0 00 002773'         JRST    VAL11X                  ;Take the +1 error return
002767' 260 17 0 00 003144' VAL11A: GO      CBLKAV                  ;Get an available cblk
002770' 332 00 0 00 000001          SKIPE   AC1                     ;Any resources available?
002771' 254 00 0 00 002775'         JRST    VAL11S                  ;Yes, continue
002772' 201 00 0 00 000375          MOVEI   AC0,NORESO              ;No, do no resources
                                        
002773' 260 17 0 00 003450' VAL11X: GO      LDSTUS                  ;Load the status byte in the rblk
002774' 334 00 0 00 000000          SKIPA                           ;Take the +1 error return
002775' 350 00 0 17 777775  VAL11S: AOS     -3(P)                   ;+2 Return
002776' 262 17 0 00 000002          GET     AC2
002777' 262 17 0 00 000001          GET     AC1
003000' 262 17 0 00 000000          GET     AC0
003001' 263 17 0 00 000000          RTN                             ;Return
```

```
2178
2179                                    ;Do the function the ctp request packet demanded
2180
2181  003002' 261 17 0 00 000000  ACT11:  PUT     AC0                     ;Save ac's
2182  003003' 261 17 0 00 000001          PUT     AC1
2183  003004' 261 17 0 00 000002          PUT     AC2
2184  003005' 402 00 0 00 000001          SETZM   AC1                     ;Don't use cid
2185  003006' 200 02 0 00 000232'         MOVE    AC2,OTHERN              ;Use node number
2186  003007' 260 17 0 00 005737'         GO      CONFIG                  ;Does node exists?
2187  003010' 254 00 0 00 003020'         JRST    ACT11B                  ;No, node unkown to sca
2188
2189  003011' 260 17 0 00 005055'         GO      CONECT                  ;Try and connect to othernode
2190  003012' 200 02 0 00 000276'         MOVE    AC2,CUREXE              ;Get rhblk
2191  003013' 506 01 0 02 000000          HRLM    AC1,SIXCID(AC2)         ;Store 6 bit cid in rhblk(6b cid,,flink)
2192  003014' 260 17 0 00 004225'         GO      EXWAIT                  ;Create a fork to wait for the response
2193  003015' 254 00 0 00 003023'         JRST    ACT11X                  ;Ok, take +1 good return
2194  003016' 201 00 0 00 000375          MOVEI   AC0,NORESO              ;Cfork failed, send back resource failure
2195  003017' 334 00 0 00 000000          SKIPA                           ;Skip next instruction
2196  003020' 201 00 0 00 000370  ACT11B: MOVEI   AC0,NODEUN               ;Yes, do node unknown to sca
2197  003021' 260 17 0 00 003450'         GO      LDSTUS                  ;Load the status byte in the rblk
2198  003022' 350 00 0 17 777775  ACT11S: AOS     -3(P)                   ;+2 Error return
2199  003023' 262 17 0 00 000002  ACT11X: GET     AC2                     ;Restore ac's
2200  003024' 262 17 0 00 000001          GET     AC1
2201  003025' 262 17 0 00 000000          GET     AC0
2202  003026' 263 17 0 00 000000          RTN                             ;Error return
```

```
                                        SUBTTL  CTP Disconnect

                                ;This routine does an orderly shutdown of a virtual circuit(connection).

003027' 332 00 0 00 000216'     LISDIS: TRMSG   (Opcode: Disconnect)
003030' 260 17 0 00 261240
003031' 261 17 0 00 000000              PUT     AC0                     ;Save ac's
003032' 260 17 0 00 003543'             GO      CHGOPC                  ;Change opcode from a req to a res.
003033' 201 00 0 00 000000              MOVEI   AC0,SUCCES              ;Use the success status
003034' 260 17 0 00 003450'             GO      LDSTUS                  ;Load status byte in the ctp packet
003035' 661 16 0 00 000002              TLO     CURSTS,(DONE)           ;Set the done bit
003036' 621 16 0 00 000001              TLZ     CURSTS,(GOBIT)          ;Clear the gobit
003037' 200 01 0 00 000276'             MOVE    AC1,CUREXE              ;Get the rhblk
003040' 202 16 0 01 000002              MOVEM   CURSTS,STATUS(AC1)      ;Store the status
003041' 260 17 0 00 004124'             GO      SEND                    ;Send the ctp response message
003042' 254 00 0 00 003044'             JRST    LISDI1                  ;Return for virtual circuit already gone
003043' 260 17 0 00 003046'             GO      ACT12                   ;Perform action for the request
003044' 262 17 0 00 000000      LISDI1: GET     AC0
003045' 263 17 0 00 000000              RTN                             ;Return

                                ;Do the function the ctp request packet demanded

003046' 261 17 0 00 000001      ACT12:  PUT     AC1                     ;Save ac's
003047' 261 17 0 00 000002              PUT     AC2
003050' 201 02 0 00 000001              MOVEI   AC2,MATCH               ;Get the disconnect reason
003051' 200 01 0 16 000000              MOVE    AC1,(CURSTS)            ;Get the cid
003052' 260 17 0 00 005314'             GO      DISCON                  ;Close this connection
003053' 262 17 0 00 000002      ACT12X: GET     AC2
003054' 262 17 0 00 000001              GET     AC1
003055' 263 17 0 00 000000              RTN                             ;Return
```

```
                                        SUBTTL  Subroutines

                                ;Get an available request header block.
                                ;This routine sees if any request header blocks are available to use.
                                ;If there is, ac1 will have the rhblk address.
                                ;Call:
                                ;       Go      rhblkv                  ;call this routine
                                ;       +1                              ;Ac1 has the available rhblk

003056' 261 17 0 00 000002      RHBLKV: PUT     AC2
003057' 261 17 0 00 000003              PUT     AC3
003060' 402 00 0 00 000003              SETZM   AC3                     ;Clr ac
003061' 515 02 0 00 777400              HRLZI   AC2,-<COUNT*RHBPER>     ;Negative count in left half
003062' 336 00 0 03 010616'     RHBLKA: SKIPN   RH1(AC3)                ;This req hdr blk available?
003063' 254 00 0 00 003067'             JRST    RHBLKB                  ;Yes
003064' 271 03 0 00 000004              ADDI    AC3,RHBLEN              ;Increment to next req hdr blk
003065' 253 02 0 00 003062'             AOBJN   AC2,RHBLKA              ;Have we checked all the req. Blk's?

                                ;We should never get here. The connect blocks(cblkx) should run out before we
                                ;find the req hdr blks all used up.

003066' 254 00 0 00 261256              CRASH   (NRH,No req hdr blocks left)

                                ;Address of available rhblk in ac1

003067' 201 01 0 03 010616'     RHBLKB: MOVEI   AC1,RH1(AC3)            ;Get rhblk address
003070' 262 17 0 00 000003              GET     AC3
003071' 262 17 0 00 000002              GET     AC2
003072' 263 17 0 00 000000              RTN                             ;Return
```

```
2262
2263                                    ;Add a request header block to the linked list.
2264                                    ;We come here only during interrupt time.
2265                                    ;Call:
2266                                    ;       Move    ac1,[n]                 ;n = rhblk address
2267                                    ;       Go      adhblk                  ;call this routine
2268                                    ;       +1                              ;Always
2269
2270  003073' 261 17 0 00 000000    ADHBLK: PUT     AC0                     ;Save ac's
2271  003074' 261 17 0 00 000001            PUT     AC1
2272  003075' 261 17 0 00 000002            PUT     AC2
2273
2274                                    ;See if the rhblk in ac1 is already linked. If it is, exit.
2275
2276  003076' 331 00 1 00 000001            SKIPL   @AC1                    ;Is the flink word = -1?
2277  003077' 254 00 0 00 261272            CRASH   (RNQ,rhblk was never queued)
2278
2279                                    ;Set up the botexe word
2280
2281  003100' 200 02 0 00 000275'           MOVE    AC2,BOTEXE              ;Get soon to be previous rhblk
2282  003101' 202 01 0 00 000275'           MOVEM   AC1,BOTEXE              ;Store this rhblk in the bottom
2283
2284                                    ;Now set up the blink and flink word in this rhblk
2285
2286  003102' 202 [illegible] 0 01 000001   MOVEM   AC2,BLINK(AC1)          ;Store previous rblk to blink
2287  003103' 200 [illegible] 0 02 000000   MOVE    AC0,FLINK(AC2)          ;Get previous rhblks flink and
2288  003104' 202 00 0 01 000000            MOVEM   AC0,FLINK(AC1)          ; Store it in this rhblks flink
2289
2290                                    ;Set up the previous rhblks flink word
2291
2292  003105' 202 01 0 02 000000            MOVEM   AC1,FLINK(AC2)          ;Store this rhblk into previous rhblks flink
2293  003106' 262 17 0 00 000002    ADHBLA: GET     AC2                     ;Restore ac's
2294  003107' 262 17 0 00 000001            GET     AC1
2295  003110' 262 17 0 00 000000            GET     AC0
2296  003111' 263 17 0 00 000000            RTN                             ;Return
2297
```

```
                                ;Remove a request header block from the linked list and make it available.
                                ;We enter this subroutine with the address of the rhblk to be removed in ac1.
                                ;Call:
                                ;       Move    ac1,[n]                 ;n = rhblk address
                                ;       Go      rmhblk                  ;call this routine
                                ;       +1                              ;Always

003112' 261 17 0 00 000000      RMHBLK: PUT     AC0                     ;Save ac's
003113' 261 17 0 00 000001              PUT     AC1
003114' 261 17 0 00 000002              PUT     AC2
003115' 261 17 0 00 000003              PUT     AC3

                                ;See if the rhblk in ac1 is already available. If it is, exit.

003116' 336 00 1 00 000001              SKIPN   @AC1                    ;Is the flink word = 0?
003117' 254 00 0 00 003137'             JRST    RMRBLB                  ;Yes, rhblk is available

                                ;Make this rhblk available to use by updating the forward req hdr blk's blink...

003120' 200 00 0 01 000001              MOVE    AC0,BLINK(AC1)          ;Get address of previous rblk
003121' 200 03 0 01 000000              MOVE    AC3,FLINK(AC1)          ;Get forward link address
003122' 202 00 0 03 000001              MOVEM   AC0,BLINK(AC3)          ;Update the blink word in the forward rhblk

                                ;..And the previous req hdr blk's flink.

003123' 200 00 0 01 000000              MOVE    AC0,FLINK(AC1)          ;Get forward link address
003124' 200 03 0 01 000001              MOVE    AC3,BLINK(AC1)          ;Get address of previous rhblk
003125' 202 00 0 03 000000              MOVEM   AC0,FLINK(AC3)          ;Update the flink word in the previous rhblk

                                ;Clear the rhblk(req1 or req2) entry in the cblk.

003126' 550 02 0 01 000002              HRRZ    AC2,STATUS(AC1)         ;Get the cblk
003127' 316 01 0 02 000002              CAMN    AC1,REQ1(AC2)           ;Are the 'RHBLKS'' the same?
003130' 402 00 0 02 000002              SETZM   REQ1(AC2)               ;Yes, clear it
003131' 316 01 0 02 000003              CAMN    AC1,REQ2(AC2)           ;Are these 'RHBLKS'' the same?
003132' 402 00 0 02 000003              SETZM   REQ2(AC2)               ;Yes, clear it

                                ;Now zero out the flink and the blink words in the req hdr blk(saying i am now available).

003133' 201 02 0 00 000003              MOVEI   AC2,<RHBLEN-1>          ;Get length of rhblk -1
003134' 402 00 1 00 000001      RMRBLA: SETZM   @AC1                    ;Clear this location
003135' 350 00 0 00 000001              AOS     AC1                     ;Increment to next location
003136' 367 02 0 00 003134'             SOJG    AC2,RMRBLA              ;Do it all

003137' 262 17 0 00 000003      RMRBLB: GET     AC3                     ;Restore ac's
003140' 262 17 0 00 000002              GET     AC2
003141' 262 17 0 00 000001              GET     AC1
003142' 262 17 0 00 000000              GET     AC0
003143' 263 17 0 00 000000              RTN                             ;Return
```

```
                                        ;Get an available connection block.
                                        ;This routine sees if any connection blocks are available to use.
                                        ;If there is, ac1 will have the cblk address.
                                        ;Call:
                                        ;       Go      cblkav                  ;call this routine
                                        ;       +1                              ;Ac1 now has the available cblk
                                        ;                                       ;Ac1 will be = to 0 if none are available

003144' 261 17 0 00 000002      CBLKAV: PUT     AC2                     ;Save ac's
003145' 261 17 0 00 000003              PUT     AC3
003146' 261 17 0 00 000004              PUT     AC4
003147' 402 00 0 00 000001              SETZM   AC1                     ;Zero out
003150' 476 00 0 00 000002              SETOM   AC2                     ;Init index pointer
003151' 201 03 0 00 000040              MOVEI   AC3,CBLSTE              ;Get size of cblst
003152' 350 00 0 00 000002      CBLKAA: AOS     AC2                     ;Increment index
003153' 200 04 0 02 010556'             MOVE    AC4,CBLST(AC2)          ;Get cblk address
003154' 332 00 0 04 000000              SKIPE   CID(AC4)                ;Is this cblk available?
003155' 367 03 0 00 003152'             SOJG    AC3,CBLKAA              ;No, keep looking
003156' 332 00 0 00 000003              SKIPE   AC3                     ;Have we finished looking?
003157' 200 01 0 00 000004              MOVE    AC1,AC4                 ;No, get cblk

                                        ;The address of the available cblk is now in ac1. If ac1 = 0 no cblks are avail

003160' 262 17 0 00 000004              GET     AC4                     ;Restore acs
003161' 262 17 0 00 000003              GET     AC3
003162' 262 17 0 00 000002              GET     AC2
003163' 263 17 0 00 000000              RTN                             ;Return
```

```
2378
2379
2380                                    ;This subroutine checks to see if any connection blocks should be removed
2381                                    ;because of a virtual circuit going away or a ctp disconnect command.  If there
2382                                    ;are, this cblk is made available along with any rblks that are connected to
2383                                    ;this cblk.
2384
2385  003164'                           RMCBLK:
2386                                    ;(12)   GO      DISINT                  ;Disable the interrupt system
2387  003164' 261 17 0 00 000000                PUT     AC0                     ;Save ac's
2388  003165' 261 17 0 00 000001                PUT     AC1
2389  003166' 261 17 0 00 000002                PUT     AC2
2390  003167' 261 17 0 00 000003                PUT     AC3
2391  003170' 261 17 0 00 000004                PUT     AC4
2392  003171' 261 17 0 00 000005                PUT     AC5
2393  003172' 476 00 0 00 000002                SETOM   AC2                     ;Init index pointer
2394  003173' 201 03 0 00 000040                MOVEI   AC3,CBLSTE              ;Get size of cblst
2395  003174' 350 00 0 00 000002        RMCBLA: AOS     AC2                     ;Increment index
2396  003175' 200 04 0 02 010556'              MOVE    AC4,CBLST(AC2)          ;Get cblk address
2397  003176' 336 00 0 04 000000                SKIPN   CID(AC4)                ;Is this cblk already available?
2398  003177' 254 00 0 00 003233'               JRST    RMCBLC                  ;Yes, look for another
2399  003200' 200 01 0 04 000001                MOVE    AC1,CBSTS(AC4)          ;Get connection block status
2400  003201' 607 01 0 00 000010                TLNN    AC1,(VCGONE)            ;Virtual circuit gone?
2401  003202' 254 00 0 00 003233'               JRST    RMCBLC                  ;No, keep looking
2402
2403                                    ;Unmap any buffers from sca that are mapped to this cblk.
2404
2405  003203' 336 01 0 04 000006                SKIPN   AC1,BFSCA1(AC4)         ;A buffer mapped?
2406  003204' 254 00 0 00 003212'               JRST    RMCBLG                  ;No, continue
2407  003205' 260 17 0 00 006040'               GO      RMVBUF                  ;Unmap this buffer
2408  003206' 255 00 0 00 000000                JFCL                            ;Good return
2409  003207' 255 00 0 00 000000                JFCL                            ;Error return
2410  003210' 200 01 0 04 000007                MOVE    AC1,BUFF1(AC4)          ;Get buffer(bufx) address
2411  003211' 402 00 1 00 000001                SETZM   @AC1                    ;Make it available
2412  003212' 336 01 0 04 000010        RMCBLG: SKIPN   AC1,BFSCA2(AC4)         ;A buffer mapped?
2413  003213' 254 00 0 00 003221'               JRST    RMCBLH                  ;No, continue
2414  003214' 260 17 0 00 006040'               GO      RMVBUF                  ;Unmap this buffer
2415  003215' 255 00 0 00 000000                JFCL                            ;Good return
2416  003216' 255 00 0 00 000000                JFCL                            ;Error return
2417  003217' 200 01 0 04 000011                MOVE    AC1,BUFF2(AC4)          ;Get buffer(bufx) address
2418  003220' 402 00 1 00 000001                SETZM   @AC1                    ;Make it available
```

```
                                ;Get rid of this cblk and any rhblks connected to it.
                                ;Remove the 1st rhblk, connected to this cblk, from the linked list.
003221' 550 01 0 04 000002      RMCBLH: HRRZ    AC1,REQ1(AC4)           ;Get the address of the 1st request
003222' 332 00 0 00 000001              SKIPE   AC1                     ;Skip if there is no rhblk
003223' 260 17 0 00 003112'             GO      RMHBLK                  ;Remove this rhblk from data structure
                                ;Remove the 2nd rblk, connected to this cblk, from the linked list.
003224' 550 01 0 04 000003      RMCBLM: HRRZ    AC1,REQ2(AC4)           ;Get the address of the 2nd request
003225' 332 00 0 00 000001              SKIPE   AC1                     ;Skip if there is no rhblk
003226' 260 17 0 00 003112'             GO      RMHBLK                  ;Remove this rhblk from data structure
                                ;Clear the cblk,making it available.
003227' 201 01 0 00 000012              MOVEI   AC1,CBKLEN              ;Get length of cblk
003230' 402 00 1 00 000004      RMCBLB: SETZM   @AC4                    ;Clear this location
003231' 350 00 0 00 000004              AOS     AC4                     ;Inc to next location
003232' 367 01 0 00 003230'             SOJG    AC1,RMCBLB              ;Do it all
003233' 367 03 0 00 003174'     RMCBLC: SOJG    AC3,RMCBLA              ;Keep looking for unwanted cblks
003234' 262 17 0 00 000005              GET     AC5                     ;Restore ac's
003235' 262 17 0 00 000004              GET     AC4
003236' 262 17 0 00 000003              GET     AC3
003237' 262 17 0 00 000002              GET     AC2
003240' 262 17 0 00 000001              GET     AC1
003241' 262 17 0 00 000000              GET     AC0
                                ;(12)   GO      ENAINT                  ;Enable the interrupt system
003242' 2o5 17 0 00 000000              RTN                             ;Return
```

```
2450
2451                                    ;Get an available dma buffer.
2452                                    ;This routine sees if any dma buffers(bufx) are available to use.
2453                                    ;If there is, ac1 will have the cblk address.
2454                                    ;Call:
2455                                    ;       Go      bufav                   ;call this routine
2456                                    ;       +1                              ;Ac1 now has the available buffer
2457                                    ;                                       ;Ac1 will be = to 0 if none are available
2458
2459  003243' 261 17 0 00 000002    BUFAV:  PUT     AC2                     ;Save ac's
2460  003244' 261 17 0 00 000003            PUT     AC3
2461  003245' 261 17 0 00 000004            PUT     AC4
2462  003246' 402 00 0 00 000001            SETZM   AC1                     ;Zero out
2463  003247' 476 00 0 00 000002            SETOM   AC2                     ;Init index pointer
2464  003250' 201 03 0 00 000040            MOVEI   AC3,BUFSTE              ;Get size of buflst
2465  003251' 350 00 0 00 000002    BUFAA:  AOS     AC2                     ;Increment index
2466  003252' 200 04 0 02 261000            MOVE    AC4,BUFLST(AC2)         ;Get bufx address
2467  003253' 332 00 1 00 000004            SKIPE   @AC4                    ;Is this bufx available?
2468  003254' 367 03 0 00 003251'           SOJG    AC3,BUFAA               ;No, keep looking
2469  003255' 332 00 0 00 000003            SKIPE   AC3                     ;Have we finished looking?
2470  003256' 200 01 0 00 000004            MOVE    AC1,AC4                 ;No, get bufx
2471
2472                                    ;The address of the available bufx is now in ac1. If ac1 = 0 no bufx are avail
2473
2474  003257' 262 17 0 00 000004            GET     AC4                     ;Restore acs
2475  003260' 262 17 0 00 000003            GET     AC3
2476  003261' 262 17 0 00 000002            GET     AC2
2477  003262' 263 17 0 00 000000            RTN                             ;Return
2478
```

```
                                ;This subroutine disconnects all 3rd party connections(to a responder) if there
                                ;aren't any connections open to a controller.  ctp states that if there isn't
                                ;any connections open to a controller all 3rd party connections should be
                                ;disconnected.  We also make all the rhblk's that were queued up at connect
                                ;time available by clearing the first word in all the rhblk's.

003263' 261 17 0 00 000000      KIL3RD: PUT     AC0                     ;Save ac's
003264' 261 17 0 00 000001              PUT     AC1
003265' 261 17 0 00 000002              PUT     AC2
003266' 261 17 0 00 000003              PUT     AC3
003267' 261 17 0 00 000004              PUT     AC4
003270' 476 00 0 00 000003              SETOM   AC3                     ;Init index pointer
003271' 201 00 0 00 000040              MOVEI   AC0,CBLSTE              ;Get size of cblst
003272' 350 00 0 00 000003      KIL3RA: AOS     AC3                     ;Increment index
003273' 200 04 0 03 010556'             MOVE    AC4,CBLST(AC3)          ;Get cblk address
003274' 336 00 0 04 000000              SKIPN   CID(AC4)                ;Do we have a connection open?
003275' 254 00 0 00 003301'             JRST    KIL3RB                  ;No, keep looking for one
003276' 200 01 0 04 000001              MOVE    AC1,CBSTS(AC4)          ;Get connection block status
003277' 607 01 0 00 200000              TLNN    AC1,(THIRD)             ;Is this a 3rd party connection?
003300' 254 00 0 00 003331'             JRST    KIL3RX                  ;No, a connection is open to a cntrlr
003301' 367 00 0 00 003272'     KIL3RB: SOJG    AC0,KIL3RA              ;Yes, this one doesn't count
```

```
2501
2502                                    ;If we fall through to here, it means there are no connections open to a
2503                                    ;controller.
2504
2505                                    ;First make all the rhblk's available that were queued at connect time.
2506
2507  003302' 402 00 0 00 000003                SETZM   AC3                     ;Clr ac
2508  003303' 515 02 0 00 777400                HRLZI   AC2,-<COUNT*RHBPER>     ;Negative count in left half
2509  003304' 332 00 0 03 010616'   KIL3RC:     SKIPE   RH1(AC3)                ;This req hdr blk available?
2510  003305' 402 00 0 03 010616'               SETZM   RH1(AC3)                ;No, make it available
2511  003306' 271 03 0 00 000004                ADDI    AC3,RHBLEN              ;Increment to next req hdr blk
2512  003307' 253 02 0 00 003304'               AOBJN   AC2,KIL3RC              ;Check all the req. Blk's
2513
2514  003310' 476 00 0 00 000003                SETOM   AC3                     ;Init index pointer
2515  003311' 201 00 0 00 000040                MOVEI   AC0,CBLSTE              ;Get size of cblst
2516  003312' 350 00 0 00 000003    KIL3RF:     AOS     AC3                     ;Increment index
2517  003313' 200 04 0 03 010556'               MOVE    AC4,CBLST(AC3)          ;Get cblk address
2518  003314' 336 00 0 04 000000                SKIPN   CID(AC4)                ;Is this cblk already available?
2519  003315' 254 00 0 00 003330'               JRST    KIL3RE                  ;Yes, look for another
2520  003316' 200 01 0 04 000001                MOVE    AC1,CBSTS(AC4)          ;Get connection block status
2521  003317' 607 01 0 00 200000                TLNN    AC1,(THIRD)             ;Is this a 3rd party connection?
2522  003320' 254 00 0 00 003330'               JRST    KIL3RE                  ;No, keep looking
2523
2524                                    ;Do a ctp "DISCONNECT" to this 3rd party responder.
2525
2526  003321' 200 01 0 04 000000                MOVE    AC1,CID(AC4)            ;Get cid in ac1
2527  003322' 201 02 0 00 000001                MOVEI   AC2,MATCH               ;Get disconnect reason in ac2
2528  003323' 260 17 0 00 005314'               GO      DISCON                  ;Do the disconnect
2529
2530                                    ;Clear the cblk,making it available.
2531
2532  003324' 201 01 0 00 000012                MOVEI   AC1,CBKLEN              ;Get length of cblk
2533  003325' 402 00 1 00 000004    KIL3RH:     SETZM   @AC4                    ;Clear this location
2534  003326' 350 00 0 00 000004                AOS     AC4                     ;Inc to next location
2535  003327' 367 01 0 00 003325'               SOJG    AC1,KIL3RH              ;Do it all
2536
2537  003330' 367 00 0 00 003312'   KIL3RE:     SOJG    AC0,KIL3RF              ;Keep looking for unwanted cblks
2538  003331' 262 17 0 00 000004    KIL3RX:     GET     AC4
2539  003332' 262 17 0 00 000003                GET     AC3
2540  003333' 262 17 0 00 000002                GET     AC2
2541  003334' 262 17 0 00 000001                GET     AC1
2542  003335' 262 17 0 00 000000                GET     AC0
2543  003336' 263 17 0 00 000000                RTN                             ;Return
```

```
2544
2545                                    ;This subroutine sends back a status of 'NOT IMPLEMENTED'' to the controller.
2546                                    ;We get here if an opcode is one that is 'NOT IMPLEMENTED'' by the responder.
2547
2548  003337' 261 17 0 00 000000    NOTIMP: PUT     AC0                     ;Save the acs
2549  003340' 261 17 0 00 000001            PUT     AC1
2550  003341' 261 17 0 00 000002            PUT     AC2
2551  003342' 200 02 0 00 000276'           MOVE    AC2,CUREXE              ;Get rblk address
2552  003343' 201 01 0 02 000003            MOVEI   AC1,CTPPKT(AC2)         ;Get  start of ctp address in ac1
2553  003344' 135 00 0 00 000313'           LDB     AC0,OPC0                ;Get opcode
2554  003345' 202 00 0 00 000231'           MOVEM   AC0,OPCODE              ;Store the original value
2555  003346' 271 00 0 00 000100            ADDI    AC0,RESOFS              ;Change opcode from a req. To a res.
2556  003347' 137 00 0 00 000313'           DPB     AC0,OPC0                ;Store it
2557  003350' 201 00 0 00 000377            MOVEI   AC0,NTIMPL              ;255 = Ctp function not implemented
2558  003351' 260 17 0 00 003450'           GO      LDSTUS                  ;Load status byte
2559  003352' 260 17 0 00 004750'           GO      SENIMP                  ;Send not implemented
2560  003353' 255 00 0 00 000000            JFCL                            ;Error return
2561  003354' 661 16 0 00 000002            TLO     CURSTS,(DONE)           ;Set the done bit
2562  003355' 621 16 0 00 000001            TL      CURSTS,(GOBIT)          ;Clear the gobit
2563  003356' 200 01 0 00 000276'           [       AC1,CUREXE              ;Get the rhblk
2564  003357' 202 16 0 01 000002            :       M   CURSTS,STATUS(AC1)  ;Store the status
2565  003360' 262 17 0 00 000002            GE      AC2
2566  003361' 262 17 0 00 000001            GET     AC1
2567  003362' 262 17 0 00 000000            GET     AC0
2568  003363' 263 17 0 00 000000            RTN                             ;Return
```

```
                                ;This routine is used to get either the repcount or actcount in its intended
                                ;format. Get bytes 6 & 7 of the ctp packet, swap the 2 bytes around and place
                                ;it right justified in ac0.
                                ;
                                ;Call:
                                ;       Go      getrep                  ;call this routine
                                ;       +1                              ;Ac0 has the corrected repcount or actcount

003364'                         GETREP:
003364' 261 17 0 00 000001      GETACT: PUT     AC1                     ;Save ac's
003365' 261 17 0 00 000002              PUT     AC2
003366' 400 00 0 00 000000              SETZ    AC0                     ;Clear it
003367' 200 01 0 00 000276'             MOVE    AC1,CUREXE              ;Get rhblk
003370' 200 02 0 01 000003              MOVE    AC2,CTPPKT(AC1)         ;Get rblk
003371' 201 01 0 02 000001              MOVEI   AC1,1(AC2)              ;Get address to  repcount word
003372' 135 01 0 00 000321'             LDB     AC1,REPC                ;Get repount in ac1
003373' 200 02 0 00 261275              MOVE    AC2,[POINT 8,AC0,27]    ;Set up pointer
003374' 137 01 0 00 000002              DPB     AC1,AC2                 ;Deposit 1st byte
003375' 242 01 0 00 777770              LSH     AC1,-^D8                ;Shift right 1 byte
003376' 136 01 0 00 000002              IDPB    AC1,AC2                 ;Deposit 2nd byte
003377' 262 17 0 00 000002              GET     AC2
003400' 262 17 0 00 000001              GET     AC1
003401' 263 17 0 00 000000              RTN

                                ;Store the repcount or actcount back in the ctp packet
                                ;Call:
                                ;       Move    ac0,[n]                 ;n = repcount or actcount
                                ;       Go      strrep                  ;call this routine
                                ;       +1                              ;Always

003402'                         STRREP:
003402' 261 17 0 00 000001      STRACT: PUT     AC1                     ;Save ac's
003403' 261 17 0 00 000002              PUT     AC2
003404' 200 01 0 00 000276'             MOVE    AC1,CUREXE              ;Get rhblk
003405' 200 02 0 01 000003              MOVE    AC2,CTPPKT(AC1)         ;Get rblk
003406' 201 01 0 02 000001              MOVEI   AC1,1(AC2)              ;Get the repcount address
003407' 200 02 0 00 261276              MOVE    AC2,[POINT 8,(AC1),23]  ;Set up pointer
003410' 137 00 0 00 000002              DPB     AC0,AC2                 ;Deposit 1st byte
003411' 242 00 0 00 777770              LSH     AC0,-^D8                ;Get the 2nd byte in position
003412' 136 00 0 00 000002              IDPB    AC0,AC2                 ;Deposit 2nd byte
003413' 262 17 0 00 000002              GET     AC2
003414' 262 17 0 00 000001              GET     AC1
003415' 263 17 0 00 000000              RTN
```

```
2614
2615                                    ;This routine is used to get the genlength in its intended format.  Get bytes
2616                                    ;12 & 13 of the ctp packet, swap the 2 bytes around and place it right
2617                                    ;justified in ac0.
2618                                    ;
2619                                    ;Call:
2620                                    ;       Go      getgln                  ;call this routine
2621                                    ;       +1                              ;Ac0 now has the corrected genlength
2622
2623  003416' 261 17 0 00 000001        GETGLN: PUT     AC1                     ;Save ac's
2624  003417' 261 17 0 00 000002                PUT     AC2
2625  003420' 400 00 0 00 000000                SETZ    AC0                     ;Clear it
2626  003421' 200 01 0 00 000276'               MOVE    AC1,CUREXE              ;Get rhblk
2627  003422' 200 02 0 01 000003                MOVE    AC2,CTPPKT(AC1)         ;Get rblk
2628  003423' 201 01 0 02 000003                MOVEI   AC1,3(AC2)              ;Get address to  genlength word
2629  003424' 135 01 0 00 000336'               LDB     AC1,GENL                ;Get genlength in ac1
2630  003425' 200 02 0 00 261275                MOVE    AC2,[POINT 8,AC0,27]    ;Set up pointer
2631  003426' 137 01 0 00 000002                DPB     AC1,AC2                 ;Deposit 1st byte
2632  003427' 242 01 0 00 777770                LSH     AC1,-^D8                ;Shift right 1 byte
2633  003430' 136 01 0 00 000002                IDPB    AC1,AC2                 ;Deposit 2nd byte
2634  003431' 262 17 0 00 000002                GET     AC2
2635  003432' 262 17 0 00 000001                GET     AC1
2636  003433' 263 17 0 00 000000                RTN
2637
2638
2639                                    ;Store the genlength back in the ctp packet
2640                                    ;Call:
2641                                    ;       Move    ac0,[n]                 ;n = genlength
2642                                    ;       Go      strgln                  ;call this routine
2643                                    ;       +1                              ;Always
2644
2645  003434' 261 17 0 00 000001        STRGLN: PUT     AC1                     ;Save ac's
2646  003435' 261 17 0 00 000002                PUT     AC2
2647  003436' 200 01 0 00 000276'               MOVE    AC1,CUREXE              ;Get rhblk
2648  003437' 200 02 0 01 000003                MOVE    AC2,CTPPKT(AC1)         ;Get rblk
2649  003440' 201 01 0 02 000003                MOVEI   AC1,3(AC2)              ;Get the genlength address
2650  003441' 200 02 0 00 261277                MOVE    AC2,[POINT 8,(AC1),7]   ;Set up pointer
2651  003442' 137 00 0 00 000002                DPB     AC0,AC2                 ;Deposit 1st byte
2652  003443' 242 00 0 00 777770                LSH     AC0,-^D8                ;Get the 2nd byte in position
2653  003444' 136 00 0 00 000002                IDPB    AC0,AC2                 ;Deposit 2nd byte
2654  003445' 262 17 0 00 000002                GET     AC2
2655  003446' 262 17 0 00 000001                GET     AC1
2656  003447' 263 17 0 00 000000                RTN
```

```
                                ;This routine loads the value that is in ac0 into the ctp packet
                                ;Status byte field contained in the rblk.
                                ;Call:
                                ;       Movei   ac0,[n]                 ;n = status value
                                ;       Go      ldstus                  ;call this routine
                                ;       +1                              ;Always
                                
003450' 261 17 0 00 000000      LDSTUS: PUT     AC0                     ;Save ac's
003451' 261 17 0 00 000001              PUT     AC1
003452' 261 17 0 00 000002              PUT     AC2
003453' 200 01 0 00 000276'             MOVE    AC1,CUREXE              ;Get rhblk
003454' 200 02 0 01 000003              MOVE    AC2,CTPPKT(AC1)         ;Get rblk
003455' 201 01 0 02 000001              MOVEI   AC1,1(AC2)              ;Get status word address
003456' 137 00 0 00 000317'             DPB     AC0,STAT                ;Load status byte
003457' 262 17 0 00 000002              GET     AC2                     ;Restore the ac's
003460' 262 17 0 00 000001              GET     AC1
003461' 262 17 0 00 000000              GET     AC0
003462' 263 17 0 00 000000              RTN
```

```
2677
2678
2679                                    ;This routine changes a 4 8 bit byte pdp11 right justified word to a 4 8 bit
2680                                    ;byte pdp10 right justified word.
2681                                    ;e.g. !Byte 0!byte 1!byte 2!byte 3!  =>  !byte 3!byte 2!byte 1!byte 0!.
2682                                    ;
2683                                    ;This routine is entered with the word to be changed in ac0.
2684                                    ;And exited with the changed word in ac0 right justified.
2685                                    ;
2686                                    ;Call:
2687                                    ;       Movei   ac0,[n]                 ;n = pdp11 word
2688                                    ;       Go      p11t10                  ;call this routine
2689                                    ;       +1                              ;Ac0 has changed word
2690
2691  003463' 261 17 0 00 000002        P11T10: PUT     AC2                     ;Save ac
2692  003464' 261 17 0 00 000003                PUT     AC3                     ;Save ac
2693  003465' 402 00 0 00 000002                SETZM   AC2
2694  003466' 200 03 0 00 261300                MOVE    AC3,[POINT 8,AC2,11]    ;Set up pointer
2695  003467' 137 00 0 00 000003                DPB     AC0,AC3                 ;Deposit 1st byte
2696  003470' 242 00 0 00 777770                LSH     AC0,-10                 ;Shift right 1 byte
2697  003471' 136 00 0 00 000003                IDPB    AC0,AC3                 ;Deposit 2nd byte
2698  003472' 242 00 0 00 777770                LSH     AC0,-10                 ;Shift right 1 byte
2699  003473' 136 00 0 00 000003                IDPB    AC0,AC3                 ;Deposit 3rd byte
2700  003474' 242 00 0 00 777770                LSH     AC0,-10                 ;Shift right 1 byte
2701  003475' 136 00 0 00 000003                IDPB    AC0,AC3                 ;Deposit 4th byte
2702  003476' 202 02 0 00 000000                MOVEM   AC2,AC0                 ;Get new number back to ac0
2703  003477' 262 17 0 00 000003                GET     AC3                     ;Restore ac
2704  003500' 262 17 0 00 000002                GET     AC2                     ;Restore ac
2705  003501' 263 17 0 00 000000                RTN                             ;Return
2706
2707                                    ;This routine changes a 4 8 bit byte pdp10 right justified
2708                                    ;word to a 4 8 bit byte pdp11 right justified word.
2709                                    ;e.g. !Byte 3!byte 2!byte 1!byte 0!  =>  !byte 0!byte 1!byte 2!byte 3!.
2710                                    ;This routine is entered with the word to be changed in ac0.
2711                                    ;And exited with the changed word in ac0 right justified.
2712                                    ;Call:
2713                                    ;       Movei   ac0,[n]                 ;n = pdp10 word
2714                                    ;       Go      p10t11                  ;call this routine
2715                                    ;       +1                              ;Ac0 has changed word
2716
2717  003502' 261 17 0 00 000002        P10T11: PUT     AC2                     ;Save ac
2718  003503' 261 17 0 00 000003                PUT     AC3                     ;Save ac
2719  003504' 402 00 0 00 000002                SETZM   AC2
2720  003505' 200 03 0 00 261300                MOVE    AC3,[POINT 8,AC2,11]    ;Set up pointer
2721  003506' 137 00 0 00 000003                DPB     AC0,AC3                 ;Deposit 1st byte
2722  003507' 242 00 0 00 777770                LSH     AC0,-10                 ;Shift right 1 byte
2723  003510' 136 00 0 00 000003                IDPB    AC0,AC3                 ;Deposit 2nd byte
2724  003511' 242 00 0 00 777770                LSH     AC0,-10                 ;Shift right 1 byte
2725  003512' 136 00 0 00 000003                IDPB    AC0,AC3                 ;Deposit 3rd byte
2726  003513' 242 00 0 00 777770                LSH     AC0,-10                 ;Shift right 1 byte
2727  003514' 136 00 0 00 000003                IDPB    AC0,AC3                 ;Deposit 4th byte
2728  003515' 202 02 0 00 000000                MOVEM   AC2,AC0                 ;Get new number back to ac0
2729  003516' 262 17 0 00 000003                GET     AC3                     ;Restore ac
2730  003517' 262 17 0 00 000002                GET     AC2                     ;Restore ac
2731  003520' 263 17 0 00 000000                RTN                             ;Return
```

```
                                        ;Check if the request received in the rblk has a valid opcode(between 0-16)
                                        ;or an opcode of 68(message) or 69(datagram).
                                        ;Call:
                                        ;       Go      opchk                   ;call this routine
                                        ;       +1                              ;Ac2 has the opcode or index number
                                        ;                                       ; 16 For a not implemented opcode.

003521' 261 17 0 00 000001      OPCHK:  PUT     AC1                     ;Save ac's
003522' 261 17 0 00 000003              PUT     AC3
003523' 200 03 0 00 000276'             MOVE    AC3,CUREXE              ;Get rhblk
003524' 200 01 0 03 000003              MOVE    AC1,CTPPKT(AC3)         ;Get rblk
003525' 135 02 0 00 000313'             LDB     AC2,OPCO                ;Get opcode
003526' 302 02 0 00 000104              CAIE    AC2,^D68                ;Opcode a generate message response?
003527' 306 02 0 00 000105              CAIN    AC2,^D69                ;Opcode a generate datagram response?
003530' 275 02 0 00 000100              SUBI    AC2,RESOFS              ;Yes, turn opcode back into a request
003531' 303 02 0 00 000020              CAILE   AC2,MAXOPC              ;The opcode greater than maximum?
003532' 201 02 0 00 000021              MOVEI   AC2,MAXOPC+1            ;Yes, invalid opcode
003533' 262 17 0 00 000003              GET     AC3
003534' 262 17 0 00 000001              GET     AC1
003535' 263 17 0 00 000000              RTN                             ;Return


                                        ;This routine fetches a unique connect id.  We don't have a problem with
                                        ;retreiving the same number twice from this routine because these 6 bits are
                                        ;only used as the high order bits in the cid that the monitor returns to us.
                                        ;
                                        ;Call:
                                        ;       Go      getcid                  ;call this routine
                                        ;       +1                              ;Ac1 has the new cid

003536' 200 01 0 00 000246'     GETCID: MOVE    AC1,CIDLOC              ;Get connect id location
003537' 350 00 0 00 000001              AOS     AC1                     ;Increment it
003540' 405 01 0 00 000077              ANDI    AC1,77                  ;6 Bits only!
003541' 202 01 0 00 000246'             MOVEM   AC1,CIDLOC              ;Store it
003542' 263 17 0 00 000000              RTN
```

```
                                ;This routine changes the opcode field in the ctp packet from a request to a
                                ;response.
                                ;
                                ;Call:
                                ;       Go      chgopc                  ;call this routine
                                ;       +1                              ;Always
                                
003543' 261 17 0 00 000000      CHGOPC: PUT     AC0                     ;Save acs
003544' 261 17 0 00 000001              PUT     AC1
003545' 261 17 0 00 000002              PUT     AC2
003546' 200 02 0 00 000276'             MOVE    AC2,CUREXE              ;Get rblk address
003547' 200 01 0 02 000003              MOVE    AC1,CTPPKT(AC2)         ;Offset to the start of the ctp packet
003550' 135 00 0 00 000313'             LDB     AC0,OPCO                ;Get opcode
003551' 303 00 0 00 000020              CAILE   AC0,MAXOPC              ;If this opcode is a
003552' 254 00 0 00 003556'             JRST    CHGOPX                  ; Response get out
003553' 202 00 0 00 000231'             MOVEM   AC0,OPCODE              ;Store it
003554' 271 00 0 00 000100              ADDI    AC0,^D64                ;Change opcode from a req. To a res.
003555' 137 00 0 00 000313'             DPB     AC0,OPCO                ;Store it
003556' 262 17 0 00 000002      CHGOPX: GET     AC2                     ;Restore acs
003557' 262 17 0 00 000001              GET     AC1
003560' 262 17 0 00 000000              GET     AC0
003561' 263 17 0 00 000000              RTN                             ;Return
```

```
                                        ;Get the return path status for a node. Call this subroutine with the node
                                        ;number of the node in ac0. Call via "GO GPRS".  The return path status will
                                        ;be returned in ac0.

003562' 261 17 0 00 000002      GPRS:   PUT     AC2                     ;Save some acs
003563' 261 17 0 00 000001              PUT     AC1                     ;
003564' 202 00 0 00 003602'             MOVEM   AC0,GPRSB+.SQRPN        ;Put the node number in argument block
003565' 402 00 0 00 003603'             SETZM   GPRSB+.SQRPS            ;Zero the last returned path status
003566' 201 01 0 00 000003              MOVEI   AC1,.LBRPS              ;Load length of block
003567' 202 01 0 00 003601'             MOVEM   AC1,.SQLEN+GPRSB        ;Save it there
003570' 201 01 0 00 000036              MOVEI   AC1,.SSRPS              ;Function code
003571' 201 02 0 00 003601'             MOVEI   AC2,GPRSB               ;Argument block address
003572' 260 17 0 00 004605'             GO      DOSCS                   ;Do the SCS JSYS or UUO
003573' 254 00 0 00 003576'              JRST   GPRS1                   ;If scs fails
003574' 200 00 0 00 003603'             MOVE    AC0,GPRSB+.SQRPS        ;Get returned path status
003575' 350 00 0 17 777776              AOS     -2(P)                   ;Bump for good return
003576' 262 17 0 00 000001      GPRS1:  GET     AC1                     ;Restore acs
003577' 262 17 0 00 000002              GET     AC2                     ;
003600' 263 17 0 00 000000              RTN                             ;Return to the code that called this

                                        ;.SSRPS argument block

003601' 000000  000003          GPRSB:  .LBRPS                          ;Length of block
003602' 000 00 0 00 000000              Z                               ;Node number
003603' 000 00 0 00 000000              Z                               ;Path status
```

```
                                ;This subroutine uses the supplied rhblk or finds an available rhblk.
                                ;Then queues it for message reception or datagram reception and sets
                                ;up the following in the rhblk.
                                ;Interrupts are shut off in this routine.
                                ;       Rhblk:  -1                      ;make this block unavailable
                                ;               ?                       ;Leave the blink alone
                                ;               0,,Cblk                 ;zero the status & store the cblk
                                ;               Rblk                    ;this rblk was always here
                                ;
                                ;Call:
                                ;       Move    ac1,[n]                 ;if n=address use this rhblk, if n=0 find an rhblk
                                ;       Move    ac2,[m]                 ;m = cblk
                                ;       Move    ac3,[v]                 ;v = cid
                                ;       Move    ac4,[x]                 ;x = [0 = queue dgm, -1 = queue msg]
                                ;       Go      queue                   ;call this routine
                                ;       +1                              ;Always

003604' 261 17 0 00 000001      QUEUE:  PUT     AC1
003605' 261 17 0 00 000002              PUT     AC2
003606' 261 17 0 00 000003              PUT     AC3
003607' 261 17 0 00 000004              PUT     AC4
                                ;(12)   GO      DISINT                  ;Disable the interrupt system
003610' 336 00 0 00 000001              SKIPN   AC1                     ;Did we get passed an rhblk to queue?
003611' 260 17 0 00 003056'             GO      RHBLKV                  ;No, get a request header block(rhblk)
003612' 476 00 0 01 000000              SETOM   FLINK(AC1)              ;Make this rhblk not available
003613' 202 02 0 01 000002              MOVEM   AC2,STATUS(AC1)         ;Store the cblk & clear the status in the rhblk

                                ;Get cid in ac1 and rblk in ac2

003614' 200 02 0 01 000003              MOVE    AC2,CTPPKT(AC1)         ;Get rblk address
003615' 200 01 0 00 000003              MOVE    AC1,AC3                 ;Need the cid in ac1
003616' 402 00 1 00 000002              SETZM   @AC2                    ;0 to Rblk, just queue this one buffer.
003617' 336 00 0 00 000004              SKIPN   AC4                     ;Queue a message?
003620' 260 17 0 00 005370'             GO      QBDGM                   ;Queue buffer for dgrm reception
003621' 332 00 0 00 000004              SKIPE   AC4                     ;Queue a message?
003622' 260 17 0 00 005342'             GO      QBMSG                   ;Queue buffer for message reception
                                ;(12)   GO      ENAINT                  ;Enable the interrupt system
003623' 262 17 0 00 000004              GET     AC4
003624' 262 17 0 00 000003              GET     AC3
003625' 262 17 0 00 000002              GET     AC2
003626' 262 17 0 00 000001              GET     AC1
003627' 263 17 0 00 000000              RTN
```

```
                                        ;This subroutine requeues the rblk that has just completed its ctp function.
                                        ;It is requeued for either datagram or message reception.
                                        ;This routine is called during interrupt time only.
                                        ;       Move    ac1,[n]                 ;n = finished rhblk
                                        ;       Move    ac2,[m]                 ;m = status,,cblk
                                        ;       Move    ac3,[x]                 ;x = cid
                                        ;       Go      quebuf                  ;call this routine
                                        ;       +1                              ;Always

003630' 261 17 0 00 000000      QUEBUF: PUT     AC0
003631' 261 17 0 00 000001              PUT     AC1
003632' 261 17 0 00 000002              PUT     AC2
003633' 261 17 0 00 000003              PUT     AC3
003634' 261 17 0 00 000004              PUT     AC4
003635' 402 00 0 00 000004              SETZM   AC4                     ;Clear datagram-message flag
003636' 607 02 0 00 000020              TLNN    AC2,(DGM)               ;Was this ctp packet received as a dgm?
003637' 476 00 0 00 000004              SETOM   AC4                     ;No, set flag to queue a msg buffer
003640' 550 02 0 00 000002              HRRZ    AC2,AC2                 ;Status,,cblk to 0,,cblk
003641' 260 17 0 00 003604'             GO      QUEUE                   ;Requeue this buffer
003642' 262 17 0 00 000004      QUEBUX: GET     AC4
003643' 262 17 0 00 000003              GET     AC3
003644' 262 17 0 00 000002              GET     AC2
003645' 262 17 0 00 000001              GET     AC1
003646' 262 17 0 00 000000              GET     AC0
003647' 263 17 0 00 000000              RTN
```

```
                                        ;See if we have a 3rd party virtual circuit open to this node already.
                                        ;Call:
                                        ;       Go      chkvcs                  ;call this routine
                                        ;       +1                              ;A vc already exists to this node(3rd party)
                                        ;       +2                              ;No vc exits

003650' 261 17 0 00 000001      CHKVCS: PUT     AC1                     ;Save ac's
003651' 261 17 0 00 000002              PUT     AC2
003652' 261 17 0 00 000003              PUT     AC3
003653' 261 17 0 00 000004              PUT     AC4
003654' 261 17 0 00 000005              PUT     AC5
003655' 402 00 0 00 000001              SETZM   AC1                     ;Zero out
003656' 476 00 0 00 000002              SETOM   AC2                     ;Init index pointer
003657' 201 03 0 00 000040              MOVEI   AC3,CBLSTE              ;Get size of cblst
003660' 350 00 0 00 000002      CHKVCA: AOS     AC2                     ;Increment index
003661' 200 04 0 02 010556'             MOVE    AC4,CBLST(AC2)          ;Get cblk address
003662' 200 05 0 04 000001              MOVE    AC5,CBSTS(AC4)          ;Get the cblk status word
003663' 607 05 0 00 200000              TLNN    AC5,(THIRD)             ;Is this cblk a 3rd party connection?
003664' 367 03 0 00 003660'     CHKVCB: SOJG    AC3,CHKVCA              ;No, keep looking
003665' 336 00 0 00 000003              SKIPN   AC3                     ;Have we checked the whole table?
003666' 254 00 0 00 003677'             JRST    CHKVCT                  ;Yes, no connection exists yet - +2 rtn

                                        ;This is a 3rd party connection. Now see if it is the same port and node
                                        ;number.

003667' 550 05 0 04 000004              HRRZ    AC5,NODE(AC4)           ;Node number to ac5
003670' 312 05 0 00 000232'             CAME    AC5,OTHERN              ;Is it the same node?
003671' 254 00 0 00 003664'             JRST    CHKVCB                  ;No, keep looking
003672' 550 00 0 04 000005              HRRZ    AC0,LOCPRT(AC4)         ;Now get the port number
003673' 550 05 0 00 000016              HRRZ    AC5,CURSTS              ;Get the "CONTROLLERS" port number
003674' 312 00 0 05 000005              CAME    AC0,LOCPRT(AC5)         ;Is it on the same ci?
003675' 254 00 0 00 003664'             JRST    CHKVCB                  ;No, keep looking
003676' 254 00 0 00 003700'             JRST    CHKVCX                  ;Yes, a connection already exists
003677' 350 00 0 17 777773      CHKVCT: AOS     -5(P)                   ;Take +2 rtn - no connection yet
003700' 262 17 0 00 000005      CHKVCX: GET     AC5                     ;Restore ac's
003701' 262 17 0 00 000004              GET     AC4
003702' 262 17 0 00 000003              GET     AC3
003703' 262 17 0 00 000002              GET     AC2
003704' 262 17 0 00 000001              GET     AC1
003705' 263 17 0 00 000000              RTN                             ;Error return
```

```
2927
2928                                    ;Whenever a connection is made, including 3rd party connections, the ''GTINFO''
2929                                    ;bit is set in the corresponding cblk by the interrupt routine. This bit being
2930                                    ;set causes this subroutine to store information in the cblk, increment an
2931                                    ;entry in the spear table and do 3rd party set up. If third party, set the
2932                                    ;''CONT'' Bit in the rhblk that initiated the connection.
2933                                    ;
2934                                    ;Call:
2935                                    ;       Go      getinf                  ;call this routine
2936                                    ;       +1                              ;Always
2937
2938  003706' 261 17 0 00 000000    GETINF: PUT     AC0                     ;Save ac's
2939  003707' 261 17 0 00 000001            PUT     AC1
2940  003710' 261 17 0 00 000002            PUT     AC2
2941  003711' 261 17 0 00 000003            PUT     AC3
2942  003712' 261 17 0 00 000004            PUT     AC4
2943  003713' 261 17 0 00 000005            PUT     AC5
2944  003714' 261 17 0 00 000006            PUT     AC6
2945  003715' 261 17 0 00 000007            PUT     AC7
2946  003716' 476 00 0 00 000005            SETOM   AC5                     ;Init index pointer
2947  003717' 201 03 0 00 000040            MOVEI   AC3,CBLSTE              ;Get size of cblst
2948  003720' 350 00 0 00 000005    GETINA: AOS     AC5                     ;Increment index
2949  003721' 200 04 0 05 010556'           MOVE    AC4,CBLST(AC5)          ;Get cblk address
2950  003722' 336 00 0 04 000000            SKIPN   CID(AC4)                ;Is this cblk available?
2951  003723' 254 00 0 00 003773'           JRST    GETINB                  ;Yes, look for another
2952  003724' 200 01 0 04 000001            MOVE    AC1,CBSTS(AC4)          ;Get connection block status
2953  003725' 607 01 0 00 100000            TLNN    AC1,(GTINFO)            ;Get information for this cblk?
2954  003726' 254 00 0 00 003773'           JRST    GETINB                  ;No, keep looking
2955
2956                                    ;Get the remote node number and store it in the cblk.
2957
2958  003727' 200 01 0 04 000000            MOVE    AC1,CID(AC4)            ;Get the cid in ac1
2959  003730' 260 17 0 00 005555'           GO      STPOLL                  ;Find the remote node # for this cblk
2960  003731' 201 01 0 00 005626'           MOVEI   AC1,STPARG              ;Get address of argument block
2961  003732' 200 00 0 01 000005            MOVE    AC0,.SQSBI(AC1)         ;Get it
2962  003733' 202 00 0 04 000004            MOVEM   AC0,NODE(AC4)           ;Node,,0 to cblk
2963
2964                                    ;Now get our port number.
2965
2966  003734' 200 01 0 04 000000            MOVE    AC1,CID(AC4)            ;Get the cid in ac1
2967  003735' 402 00 0 00 000002            SETZM   AC2                     ;Don't use the node #
2968  003736' 260 17 0 00 005737'           GO      CONFIG                  ;Find the port # for this cblk
2969  003737' 255 00 0 00 000000            JFCL                            ;Error - unlikely
2970  003740' 201 01 0 00 005767'           MOVEI   AC1,CNFARG              ;Get address of argument block
2971  003741' 200 00 0 01 000023            MOVE    AC0,.SQLPN(AC1)         ;Get it
2972  003742' 202 00 0 04 000005            MOVEM   AC0,LOCPRT(AC4)         ;Port# to cblk
```

!

```
                                ;Increment the spear table for this connection.

003743' 221 00 0 00 000020              IMULI   AC0,SPOFST              ;Create an index number
003744' 270 01 0 00 000000              ADD     AC1,AC0                 ; For the spear table
003745' 200 00 0 00 261301              MOVE    AC0,[1,,1]              ;Increment both halves
003746' 272 00 0 01 007341'             ADDM    AC0,SPEART(AC1)         ;Increment # of connections on this node

                                ;Clear the get information flag.

003747' 200 01 0 04 000001              MOVE    AC1,CBSTS(AC4)          ;Get connection block status
003750' 621 01 0 00 100000              TLZ     AC1,(GTINFO)            ;Reset the get information bit
003751' 202 01 0 04 000001              MOVEM   AC1,CBSTS(AC4)          ;Set connection block status

                                ;If this cblk is 3rd party go on, if not look for another cblk that needs info.

003752' 607 01 0 00 200000              TLNN    AC1,(THIRD)             ;3Rd party?
003753' 254 00 0 00 003773'             JRST    GETINB                  ;No, look for another cblk

                                ;Get the 6 left most bits of the cblk's cid so we can find the rhblk.

003754' 554 07 0 04 000000              HLRZ    AC7,CID(AC4)            ;Get the left half of the cid
003755' 242 07 0 00 777764              LSH     AC7,-^D12               ;Right justify the 6 bit cid

                                ;Now find the rhblk that was responsible for making this connection
                                ;And set its (cont) bit to flag that the connection was successful.

003756' 402 00 0 00 000001      GETIN:  SETZM   AC1                     ;Clr ac
003757' 515 02 0 00 777400              HRLZI   AC2,-<COUNT*RHBPER>     ;Negative count in left half
003760' 201 06 0 01 010616'     GETIND: MOVEI   AC6,RH1(AC1)            ;Get rhblk address
003761' 554 00 0 06 000000              HLRZ    AC0,FLINK(AC6)          ;Get the 6 bit cid
003762' 312 00 0 00 000007              CAME    AC0,AC7                 ;Are the cid's equal?
003763' 254 00 0 00 003771'             JRST    GETINE                  ;No, keep looking
003764' 200 01 0 06 000002              MOVE    AC1,STATUS(AC6)         ;Get the status word
003765' 661 01 0 00 000200              TLO     AC1,(CONT)              ;3Rd party connection was successful
003766' 202 01 0 06 000002              MOVEM   AC1,STATUS(AC6)         ;Restore the status word
003767' 260 17 0 00 004301'             GO      CANTIM                  ;Cancel any timer interrupt
003770' 254 00 0 00 003773'             JRST    GETINB                  ;See if anymore cblks want info.
003771' 271 01 0 00 000004      GETINE: ADDI    AC1,RHBLEN              ;Increment to next req hdr blk
003772' 253 02 0 00 003760'             AOBJN   AC2,GETIND              ;Have we checked all the req. Blk's?
003773' 367 03 0 00 003720'     GETINB: SOJG    AC3,GETINA              ;See if other cblks need info
003774' 262 17 0 00 000007      GETINX: GET     AC7                     ;Restore ac's
003775' 262 17 0 00 000006              GET     AC6
003776' 262 17 0 00 000005              GET     AC5
003777' 262 17 0 00 000004              GET     AC4
004000' 262 17 0 00 000003              GET     AC3
004001' 262 17 0 00 000002              GET     AC2
004002' 262 17 0 00 000001              GET     AC1
004003' 262 17 0 00 000000              GET     AC0
004004' 263 17 0 00 000000              RTN                             ;Return
```

```
3023
3024                                    ;This subroutine checks to see if we should make a spear entry.
3025                                    ;Check to see if we have waited the allotted time. If we have,
3026                                    ;Make an entry into spear with the following information:
3027                                    ;       "TIME OF THIS ENTRY"
3028                                    ;       "PROGRAM NAME(DFCIC)"
3029                                    ;       "VERSION NUMBER"
3030                                    ;       "TIME OF LAST ENTRY"
3031                                    ;       "PORT NUMBER"
3032                                    ;       "NODE NUMBER"
3033                                    ;       "TOTAL NUMBER OF CONNECTIONS"
3034                                    ;       "NUMBER OF CONNECTIONS SINCE LAST ENTRY"
3035                                    ;Call:
3036                                    ;       Go      spear                   ;call this routine
3037                                    ;       +1                              ;Always
3038
3039  004005' 336 00 0 00 000234'       SPEAR:  SKIPN   MONTYP                  ;Skip if TOPS-orange
3040  004006' 263 17 0 00 000000              RTN                             ;TOPS-blue, return now
3041  004007' 261 17 0 00 000001              PUT     AC1                     ;Save some acs
3042  004010' 261 17 0 00 000002              PUT     AC2
3043  004011' 261 17 0 00 000003              PUT     AC3
3044  004012' 261 17 0 00 000004              PUT     AC4
3045  004013' 260 17 0 00 004507'              GO      UTIME                   ;Get time in milliseconds
3046  004014' 231 01 0 00 135600              IDIVI   AC1,^D1000*60           ;Change ms to minutes
3047  004015' 200 02 0 00 000254'              MOVE    AC2,TIMESP              ;Get the last time we did a spear entry
3048  004016' 271 02 0 00 000074              ADDI    AC2,TIMELN              ;Add time length(min)
3049  004017' 336 00 0 00 000217'              SKIPN   SPEARF                  ;Made any spear entries ?
3050  004020' 254 00 0 00 004023'              JRST    SPEARA                  ; No, make the initial one at start up
3051  004021' 315 01 0 00 000002              CAMGE   AC1,AC2                 ;Time to do another spear entry?
3052  004022' 254 00 0 00 004026'              JRST    SPEARX                  ;No, don't do any spear entries this time
3053
3054  004023' 202 02 0 00 000254'       SPEARA: MOVEM   AC2,TIMESP              ;Store the minutes for the next time
3055  004024' 475 00 0 00 000217'              SETOM   SPEARF                  ;Mark that initial spear entry made
3056  004025' 260 17 0 00 004033'              GO      ENTER                   ;Make a spear entry
3057  004026' 262 17 0 00 000004       SPEARX: GET     AC4                     ;Restore the acs
3058  004027' 262 17 0 00 000003              GET     AC3
3059  004030' 262 17 0 00 000002              GET     AC2
3060  004031' 262 17 0 00 000001              GET     AC1
3061  004032' 263 17 0 00 000000              RTN
```

```
3062
3063                                    ;This is where we set up the syerr jsys information.
3064
3065  004033' 261 17 0 00 000001  ENTER:  PUT     AC1                     ;Save some acs
3066  004034' 261 17 0 00 000002          PUT     AC2
3067  004035' 261 17 0 00 000003          PUT     AC3
3068  004036' 261 17 0 00 000004          PUT     AC4
3069  004037' 261 17 0 00 000005          PUT     AC5
3070  004040' 261 17 0 00 000006          PUT     AC6
3071  004041' 261 17 0 00 000007          PUT     AC7
3072
3073                                    ;Get the number of entries made in the spear table(speart) and store it in the
3074                                    ;Syerr table(sysert). Also get any other information required.
3075
3076  004042' 402 00 0 00 000002          SETZM   AC2                     ;Init counter
3077  004043' 515 03 0 00 777600          HRLZI   AC3,-SPEARE             ;Get length of table
3078  004044' 336 07 0 03 007341' ENTERC: SKIPN   AC7,SPEART(AC3)         ;Is this entry empty?
3079  004045' 254 00 0 00 004060'         JRST    ENTERO                  ;Yes, keep looking
3080
3081  004046' 550 04 0 00 000003  ENTERD: HRRZ    AC4,AC3                 ;Get the index register count
3082  004047' 231 04 0 00 000020          IDIVI   AC4,SPOFST              ;Calculate the node and port number
3083  004050' 250 04 0 00 000005          EXCH    AC4,AC5                 ;Port# to ac5, node# to ac4
3084
3085  004051' 201 01 0 02 007556' ENTERE: MOVEI   AC1,SYSE12+2(AC2)       ;Get the address for storage
3086  004052' 137 05 0 00 000360'         DPB     AC5,SPRCIN              ;Store the port # away
3087
3088  004053' 201 01 0 02 007556' ENTERG: MOVEI   AC1,SYSE12+2(AC2)       ;Get the address for storage
3089  004054' 137 04 0 00 000361'         DPB     AC4,SPRNOD              ;Store the node # away
3090
3091  004055' 350 00 0 00 000002  ENTREN: AOS     AC2                     ;Increment the count
3092  004056' 202 07 0 02 007556'         MOVEM   AC7,SYSE12+2(AC2)       ;Put this entry in the sys error table
3093  004057' 350 00 0 00 000002          AOS     AC2
3094  004060' 253 03 0 00 004044' ENTERO: AOBJN   AC3,ENTERC              ;Keep looking
```

```
3095
3096                                    ;Ac2 has the number of entries made in the "SPEART" table.
3097
3098  004061' 336 00 0 00 000002                SKIPN   AC2                     ;Skip if at least one entry exists
3099  004062' 254 00 0 00 004066                JRST    ENTERS                  ; No entries exist
3100  004063' 271 02 0 00 000002                ADDI    AC2,2                   ;Add 2 for block type 12
3101  004064' 201 01 0 00 007554'                MOVEI   AC1,SYSE12              ;Get start of block type 12
3102  004065' 137 02 0 00 000357'                DPB     AC2,SPRBLN              ;Deposit the block length!
3103  004066' 271 02 0 00 000013        ENTERS: ADDI    AC2,SYSHLF              ;Add blocks 2 & 10 plus 4 header words
3104  004067' 202 02 0 00 000255'                MOVEM   AC2,SYSAC2              ;Store it for when we use the "SYERR" jsys
3105
3106                                    ;Get the "DATE/TIME" of the last spear entry(block 2) and store it in block 12.
3107
3108  004070' 200 01 0 00 007546'                MOVE    AC1,SYSE02+1            ;Get date/time
3109  004071' 202 01 0 00 007555'                MOVEM   AC1,SYSE12+1            ; And store it
3110
3111                                    ;Now store the "DATE/TIME" in block type 2(syse02+1).
3112
3113  004072' 260 17 0 00 004537'                GO      DATETM                  ;Get the date/time
3114  004073' 202 01 0 00 007546'                MOVEM   AC1,SYSE02+1            ; In ac1 and store it
3115
3116                                    ;This is where we call the jsys "SYERR".
3117
3118  004074' 260 17 0 00 004111'                GO      SYSJYS                  ;Syerr jsys
3119
3120                                    ;Clear the "NUMBER OF CONNECTIONS SINCE LAST TIME"(right half of table).
3121
3122  004075' 515 02 0 00 777600                HRLZI   AC2,-SPEARE             ;Get length of table
3123  004076' 200 01 0 02 007341'        ENTERP: MOVE    AC1,SPEART(AC2)         ;Get a spear table entry
3124  004077' 512 01 0 02 007341'                HLLZM   AC1,SPEART(AC2)         ;Clear the # of conn since last entry
3125  004100' 253 02 0 00 004076'                AOBJN   AC2,ENTERP              ;Clear the right half of the table
3126  004101' 262 17 0 00 000007                GET     AC7                     ;Restore the acs
3127  004102' 262 17 0 00 000006                GET     AC6
3128  004103' 262 17 0 00 000005                GET     AC5
3129  004104' 262 17 0 00 000004                GET     AC4
3130  004105' 262 17 0 00 000003                GET     AC3
3131  004106' 262 17 0 00 000002                GET     AC2
3132  004107' 262 17 0 00 000001                GET     AC1
3133  004110' 263 17 0 00 000000                RTN
3134
3135                                    ;This routine executes the "SYERR" jsys for spear entries.
3136
3137  004111' 336 00 0 00 000234'       SYSJYS: SKIPN   MONTYP                  ;Skip if orange
3138  004112' 263 17 0 00 000000                RTN                             ;Blue system, do nothing
3139  004113' 261 17 0 00 000001                PUT     AC1                     ;Save acs
3140  004114' 261 17 0 00 000002                PUT     AC2
3141  004115' 201 01 0 00 007541'                MOVEI   AC1,SYSERT              ;Address of argument block
3142  004116' 200 02 0 00 000255'                MOVE    AC2,SYSAC2              ;Current length of the argument block
3143  004117' 104 00 0 00 000527                SYERR%                          ;Execute the jsys
3144  004120' 320 16 0 00 004121'                 ERJMP  .+1                     ;Ignore error if any
3145  004121' 262 17 0 00 000002                GET     AC2
3146  004122' 262 17 0 00 000001                GET     AC1
3147  004123' 263 17 0 00 000000                RTN                             ;Return
```

```
3148                            ;Z:<GSCOTT.DFCIC>DFCIC2.MAC.147 27-Aug-85 11:17:39, Edit by GSCOTT
3149                            ;Add SSCBF% error code
3150                            ;Z:<GSCOTT.DFCIC>DFCIC2.MAC.142 27-Aug-85 08:24:19, Edit by GSCOTT
3151                            ;Add output of arg block address in ERRSCS output routine
3152                            ;Z:<GSCOTT.DFCIC>DFCIC2.MAC.140 23-Aug-85 15:13:58, Edit by GSCOTT
3153                            ;Add the latest SCSUUO error codes
3154                            ;Z:<GSCOTT.DFCIC>DFCIC2.MAC.139 23-Aug-85 08:50:11, Edit by GSCOTT
3155                            ;Try to catch those rare bad CPT packet pointer by doing HRRZMs
3156                            ;Z:<GSCOTT.DFCIC>DFCIC2.MAC.138  2-Aug-85 18:53:37, Edit by GSCOTT
3157                            ;Correct DIAG UUO call to use -n,,loc in the AC.
3158                            ;Z:<GSCOTT.DFCIC>DFCIC2.MAC.137  2-Aug-85 18:23:11, Edit by GSCOTT
3159                            ;Don't turn on ints until waiting for an int to happen.
3160                            ;Z:<GSCOTT.DFCIC>DFCIC2.MAC.135  2-Aug-85 16:06:43, Edit by GSCOTT
3161                            ;Turn off interrupts when processing things to do at lower level
3162                            ;Z:<GSCOTT.DFCIC>DFCIC2.MAC.134  2-Aug-85 15:55:29, Edit by GSCOTT
3163                            ;Output calling PC in DIAG UUO failures
3164                            ;Z:<GSCOTT.DFCIC>DFCIC2.MAC.129  2-Aug-85 15:15:02, Edit by GSCOTT
3165                            ;Correct output of error number on diag uuo failures.
```

```
                                        SUBTTL  Send CTP Response

                                ;Send the ctp response
                                ;This routine sets up the argument block for the scs% jsys monitor call.
                                ;Call:
                                ;       Go      send                    ;call this routine
                                ;       +1                              ;Error return
                                ;       +2                              ;Sucessful return

004124' 261 17 0 00 000001      SEND:   PUT     AC1                     ;Save the ac's
004125' 261 17 0 00 000002              PUT     AC2
004126' 261 17 0 00 000003              PUT     AC3
004127' 261 17 0 00 000004              PUT     AC4
004130' 200 01 0 16 000000              MOVE    AC1,(CURSTS)            ;Get the "CID" of this connection &
004131' 200 03 0 00 000276'             MOVE    AC3,CUREXE              ;Start of ctp
004132' 200 02 0 03 000003              MOVE    AC2,CTPPKT(AC3)         ; Request in
004133' 200 04 0 00 000231'             MOVE    AC4,OPCODE              ;Get the opcode
004134' 303 04 0 00 000020              CAILE   AC4,MAXOPC              ;Opcode a response opcode
004135' 275 04 0 00 000100              SUBI    AC4,RESOFS              ;Yes, make it a request opcode
004136' 200 03 0 04 007317'             MOVE    AC3,NMBYTS(AC4)         ;Store length
004137' 260 17 0 00 005506'             GO      SENDMD                  ;Send a dgm or msg
004140' 334 00 0 00 000000              SKIPA                           ;Error, take +1 return
004141' 350 00 0 17 777774              AOS     -4(P)                   ;Make it a +2 return
004142' 262 17 0 00 000004      SENDA:  GET     AC4                     ;Restore the ac's
004143' 262 17 0 00 000003              GET     AC3
004144' 262 17 0 00 000002              GET     AC2
004145' 262 17 0 00 000001              GET     AC1
004146' 263 17 0 00 000000              RTN                             ;Thats it
```

```
3195
3196                                    ;Send the "GENERATE MESSAGE" or "GENERATE DATAGRAM" ctp response .
3197                                    ;This routine sets up the argument block for the scs% jsys monitor call.
3198                                    ;Call:
3199                                    ;       Go      sen04                   ;call this routine
3200                                    ;       +1                              ;Error return
3201                                    ;       +2                              ;Sucessful return
3202
3203  004147' 261 17 0 00 000001    SEN04:  PUT     AC1                     ;Save the ac's
3204  004150' 261 17 0 00 000002            PUT     AC2
3205  004151' 261 17 0 00 000003            PUT     AC3
3206  004152' 261 17 0 00 000004            PUT     AC4
3207  004153' 200 01 0 16 000000            MOVE    AC1,(CURSTS)            ;Get the "CID" of this connection
3208  004154' 200 03 0 00 000276'           MOVE    AC3,CUREXE              ;Start of ctp
3209  004155' 200 02 0 03 000003            MOVE    AC2,CTPPKT(AC3)         ; Request in ac2
3210  004156' 200 03 0 00 000226'           MOVE    AC3,GENLEN              ;Get the length of the message
3211  004157' 271 03 0 00 000016            ADDI    AC3,^D14                ;Add # of bytes in packet up to image data field
3212  004160' 260 17 0 00 005506'           GO      SENDMD                  ;Send the response back
3213  004161' 334 00 0 00 000000            SKIPA                           ;Error, take +1 return
3214  004162' 350 00 0 17 777774            AOS     -4(P)                   ;Make it a +2 return
3215  004163' 262 17 0 00 000004    SEN04C: GET     AC4                     ;Restore the ac's
3216  004164' 262 17 0 00 000003            GET     AC3
3217  004165' 262 17 0 00 000002            GET     AC2
3218  004166' 262 17 0 00 000001            GET     AC1
3219  004167' 263 17 0 00 000000            RTN                             ;Thats it
```

```
3220                                            SUBTTL  Timeout Routines
3221
3222                                    ;This routine waits the amount of time stated in the delay field of a ctp
3223                                    ;request.  This is done because we want to delay only the ctp request that has
3224                                    ;asked to be delayed and not any other outstanding ctp request.
3225
3226                                    ;Call:
3227                                    ;       Go      exdly                   ;call this routine
3228                                    ;       +1                              ;The delay has been started
3229                                    ;       +2                              ;Jsys failure
3230                                    ;       +3                              ;The delay byte was = 0(no delay)
3231
3232  004170' 261 17 0 00 000000    EXDLY:  PUT     AC0                     ;Save ac's
3233  004171' 261 17 0 00 000001            PUT     AC1
3234  004172' 261 17 0 00 000002            PUT     AC2
3235  004173' 261 17 0 00 000003            PUT     AC3
3236  004174' 261 17 0 00 000004            PUT     AC4
3237  004175' 200 01 0 00 000276'           MOVE    AC1,CUREXE              ;Get rhblk
3238  004176' 200 02 0 01 000003            MOVE    AC2,CTPPKT(AC1)         ;Get rblk
3239  004177' 201 01 0 02 000001            MOVEI   AC1,1(AC2)              ;Get the delay byte word address
3240  004200' 135 00 0 00 000316'           LDB     AC0,DELA                ;Delay byte in ac0
3241  004201' 202 00 0 00 000242'           MOVEM   AC0,DLYTIM              ;Store the delay for dlytmr routine
3242  004202' 322 00 0 00 004215'           JUMPE   AC0,EXDLY2              ;Delay = 0? Yes, take the +3 return
3243  004203' 621 16 0 00 000001            TLZ     CURSTS,(GOBIT)          ;No, clear the go bit
3244
3245  004204' 201 01 0 00 004352'           MOVEI   AC1,DLYGO               ;Load address of routine to call
3246  004205' 260 17 0 00 004250'           GO      SETTIM                  ;Set a timer interrupt from time in AC0
3247  004206' 254 00 0 00 004212'            JRST   EXDLY1                  ;Give error status
3248
3249  004207' 201 00 0 00 000000            MOVEI   AC0,SUCCES              ;Zero the
3250  004210' 260 17 0 00 003450'           GO      LDSTUS                  ;  Status byte in the ctp packet
3251  004211' 254 00 0 00 004217'           JRST    EXDLY4                  ;Take the +1 return
3252
3253                                    ;Here for errors
3254
3255  004212' 201 00 0 00 000375    EXDLY1: MOVEI   AC0,NORESO              ;Send back 'RESOURCE FAILURE'
3256  004213' 260 17 0 00 003450'           GO      LDSTUS                  ;Load status byte in the ctp packet
3257  004214' 254 00 0 00 004216'           JRST    EXDLY3                  ;Take the +2 return
3258
3259  004215' 350 00 0 17 777773    EXDLY2: AOS     -5(P)                   ;+3 Return
3260  004216' 350 00 0 17 777773    EXDLY3: AOS     -5(P)                   ;+2 Return
3261  004217' 262 17 0 00 000004    EXDLY4: GET     AC4                     ;Restore ac's
3262  004220' 262 17 0 00 000003            GET     AC3
3263  004221' 262 17 0 00 000002            GET     AC2
3264  004222' 262 17 0 00 000001            GET     AC1
3265  004223' 262 17 0 00 000000            GET     AC0
3266  004224' 263 17 0 00 000000            RTN
```

```
3267
3268                                    ;This routine is used to wait for a connection response from a responder.  This
3269                                    ;routine waits a predetermined amount of time before setting the 'OVRDUE' bit
3270                                    ;in the status word of the cblk.
3271
3272                                    ;Call:
3273                                    ;       Go      exwait                  ;call this routine
3274                                    ;       +1                              ;The delay has been started
3275                                    ;       +2                              ;Cfork failure
3276
3277  004225' 261 17 0 00 000000    EXWAIT: PUT     AC0                     ;Save ac's
3278  004226' 261 17 0 00 000001            PUT     AC1
3279  004227' 261 17 0 00 000002            PUT     AC2
3280  004230' 201 00 0 00 000170            MOVEI   AC0,^D120               ;Wait 60 sec.(Each count = 500ms)
3281  004231' 201 01 0 00 004355'           MOVEI   AC1,WATOVR              ;Load routine to call later
3282  004232' 621 16 0 00 000200            TLZ     CURSTS,(CONT)           ;Now clear the continue bit
3283  004233' 260 17 0 00 004250'           GO      SETTIM                  ;Set timer for that
3284  004234' 254 00 0 00 004240'            JRST   EXWAI1                  ;Loser
3285
3286                                    ;Load the status byte with a successful code.
3287
3288  004235' 201 00 0 00 000000            MOVEI   AC0,SUCCES              ;Zero the
3289  004236' 260 17 0 00 003450'           GO      LDSTUS                  ; Status byte in the ctp packet
3290  004237' 254 00 0 00 004244'           JRST    EXWAI4                  ;Take the +1 return
3291
3292                                    ;This is the error routine
3293
3294  004240' 201 00 0 00 000375    EXWAI1: MOVEI   AC0,NORESO              ;Send back "RESOURCE FAILURE"
3295  004241' 260 17 0 00 003450'           GO      LDSTUS                  ;Load status byte in the ctp packet
3296  004242' 254 00 0 00 004243'           JRST    EXWAI3                  ;Take the +2 return
3297
3298  004243' 350 00 0 17 777775    EXWAI3: AOS     -3(P)                   ;+2 Return
3299  004244' 262 17 0 00 000002    EXWAI4: GET     AC2                     ;Restore ac's
3300  004245' 262 17 0 00 000001            GET     AC1
3301  004246' 262 17 0 00 000000            GET     AC0
3302  004247' 263 17 0 00 000000            RTN
```

```
3303
3304                                    ;Routine to cause a timer interrupt.
3305                                    ;
3306                                    ;Call:  MOVE    AC0,milliseconds_to_timer_interrupt
3307                                    ;       MOVEI   AC1,routine_to_call
3308                                    ;       GO      SETTIM
3309
3310  004250' 205 04 0 00 777740    SETTIM: MOVSI   AC4,-COUNT              ;Load -count,,0
3311  004251' 336 00 0 04 012616'   SETTI7: SKIPN   DLYADR(AC4)             ;This slot in use?
3312  004252' 254 00 0 00 004255'           JRST    SETTI6                  ;Nope, use it
3313  004253' 253 04 0 00 004251'           AOBJN   AC4,SETTI7              ;Loop for more
3314  004254' 254 00 0 00 261312            CRASH   (DBF,Delay block full)  ;Oops
3315
3316                                    ;Found an entry.  Store rhblk addr and time to wake up, then set timer
3317
3318  004255' 200 02 0 00 000276'   SETTI6: MOVE    AC2,CUREXE              ;Load the current rhblk addr
3319  004256' 202 02 0 04 012616'           MOVEM   AC2,DLYADR(AC4)         ;Save it here
3320  004257' 202 01 0 04 012716'           MOVEM   AC1,DLYRTN(AC4)         ;Save routine to call later
3321  004260' 260 17 0 00 004507'           GO      UTIME                   ;Get current uptime in milliseconds
3322  004261' 221 00 0 00 000764            IMULI   AC0,^D500               ;Convert ctp counts to millisecs
3323  004262' 270 01 0 00 000000            ADD     AC1,AC0                 ;Get time to wake this request
3324  004263' 202 01 0 04 012656'           MOVEM   AC1,DLYUPT(AC4)         ;Save the delay time
3325
3326                                    ;Now set the timer to interrupt us later and return
3327
3328  004264' 336 00 0 00 000234'           SKIPN   MONTYP                  ;Skip if TOPS-orange
3329  004265' 254 00 0 00 004275'           JRST    SETTI1                  ;TOPS-blue
3330  004266' 200 02 0 00 000000            MOVE    AC2,AC0                 ;Copy time to AC2
3331  004267' 200 01 0 00 261315            MOVE    AC1,[.FHSLF,,.TIMEL]    ;Load elapsed time till int in AC1
3332  004270' 201 03 0 00 000000            MOVEI   AC3,TIMCHN              ;Load channel to interrupt on
3333  004271' 104 00 0 00 000522            TIMER%                          ;Set the timer
3334  004272' 320 16 0 00 004274'            ERJMP  .+2                     ;Non skip return
3335  004273' 350 00 0 17 000000    SETTI2: AOS     (P)                     ;Skip return
3336  004274' 263 17 0 00 000000            RTN                             ;Return to caller
3337
3338  004275' 661 00 0 00 400000    SETTI1: TLO     AC0,(PS.TMS)            ;Set milliseconds flag
3339  004276' 047 00 0 00 000203            PITMR.  AC0,                    ;or CALLI AC,203
3340  004277' 263 17 0 00 000000             RTN                            ;Error return
3341  004300' 254 00 0 00 004273'           JRST    SETTI2                  ;Normal return
```

```
3342
3343                                    ;Routine to cancel timer interrupt.
3344                                    ;
3345                                    ;Call:  GO      CANTIM
3346
3347  004301' 261 17 0 00 000001    CANTIM: PUT     AC1                     ;Save some acs
3348  004302' 261 17 0 00 000002            PUT     AC2
3349
3350  004303' 205 01 0 00 777740            MOVSI   AC1,-COUNT              ;Load -count to look through table
3351  004304' 200 02 0 00 000276'           MOVE    AC2,CUREXE              ;Load currently executing
3352  004305' 316 02 0 01 012616'   CANTI1: CAMN    AC2,DLYADR(AC1)         ;Match?
3353  004306' 402 00 0 01 012616'           SETZM   DLYADR(AC1)             ;Yep cancel the request
3354  004307' 253 01 0 00 004305'           AOBJN   AC1,CANTI1              ;Loop for entire table
3355
3356  004310' 262 17 0 00 000002            GET     AC2                     ;Restore the acs
3357  004311' 262 17 0 00 000001            GET     AC1
3358  004312' 263 17 0 00 000000            RTN                             ;Return with request cancelled
3359
3360                                    ;Routine to handle timer interrupts on TOPS-10
3361
3362  004313' 261 17 0 00 000001    TIMR10: PUT     AC1                     ;Save AC1-AC4
3363  004314' 261 17 0 00 000002            PUT     AC2
3364  004315' 261 17 0 00 000003            PUT     AC3
3365  004316' 261 17 0 00 000004            PUT     AC4
3366
3367  004317' 260 17 0 00 004337'           GO      TIMINT                  ;Look for timer interrupt service
3368
3369  004320' 262 17 0 00 000004            GET     AC4                     ;Restore ac1-4
3370  004321' 262 17 0 00 000003            GET     AC3
3371  004322' 262 17 0 00 000002            GET     AC2
3372  004323' 262 17 0 00 000001            GET     AC1
3373  004324' 047 00 0 00 000137            DEBRK.                          ;Dismiss interrupt
3374
3375                                    ;Routine to handle timer interrupt on TOPS-20
3376
3377  004325' 261 17 0 00 000001    TIMR20: PUT     AC1                     ;Save AC1-AC4
3378  004326' 261 17 0 00 000002            PUT     AC2
3379  004327' 261 17 0 00 000003            PUT     AC3
3380  004330' 261 17 0 00 000004            PUT     AC4
3381
3382  004331' 260 17 0 00 004337'           GO      TIMINT                  ;Look for timer interrupt service
3383
3384  004332' 262 17 0 00 000004            GET     AC4                     ;Restore ac1-ac4
3385  004333' 262 17 0 00 000003            GET     AC3
3386  004334' 262 17 0 00 000002            GET     AC2
3387  004335' 262 17 0 00 000001            GET     AC1
3388  004336' 104 00 0 00 000136            DEBRK%                          ;Dismiss interrupt
```

```
                                ;Routine called at timer interrupt to set appropriate bits.

004337' 205 02 0 00 777740      TIMINT: MOVSI   AC2,-COUNT              ;Prepare to look through the table
004340' 260 17 0 00 004507'             GO      UTIME                   ;Get uptime in AC1
004341' 336 04 0 02 012616'     TIMIN1: SKIPN   AC4,DLYADR(AC2)         ;Skip if something there
004342' 254 00 0 00 004347'             JRST    TIMIN7                  ;Nothing there, check next one
004343' 315 01 0 02 012656'             CAMGE   AC1,DLYUPT(AC2)         ;Does this one need service?
004344' 254 00 0 00 004347'             JRST    TIMIN7                  ;Nope, not yet
004345' 260 17 1 02 012716'             GO      @DLYRTN(AC2)            ;Check this status
004346' 402 00 0 02 012616'             SETZM   DLYADR(AC2)             ;Clear that this one needs service
004347' 253 02 0 00 004341'     TIMIN7: AOBJN   AC2,TIMIN1              ;Loop for all of them
004350' 201 01 0 00 000004              MOVEI   AC1,TIOFFS              ;Load timer offset
004351' 254 00 0 00 004460'             JRST    CWAT1                   ;Wakeup if needed and return

                                ;Set the go bit.

004352' 205 03 0 00 000001      DLYGO:  MOVSI   AC3,(GOBIT)             ;Set the go bit
004353' 436 03 0 04 000002              IORM    AC3,STATUS(AC4)         ; In the rhblk status word
004354' 263 17 0 00 000000              RTN

                                ;Set the overdue bit.

004355' 205 03 0 00 000040      WATOVR: MOVSI   AC3,(OVRDUE)            ;Set the overdue bit
004356' 436 03 0 04 000002              IORM    AC3,STATUS(AC4)         ; In the rhblk status word
004357' 263 17 0 00 000000              RTN                             ;Return
```

```
                                        SUBTTL  Interrupt System


                                ;Set up the interrupt system.

004360' 261 17 0 00 000001      IFPINT: PUT     AC1                     ;Save the acs
004361' 261 17 0 00 000002              PUT     AC2
004362' 261 17 0 00 000003              PUT     AC3
004363' 261 17 0 00 000004              PUT     AC4
004364' 336 00 0 00 000234'             SKIPN   MONTYP                  ;Tops-20?
004365' 254 00 0 00 004402'             JRST    IFPIN1                  ;No, Tops-10
004366' 201 01 0 00 400000              MOVEI   AC1,.FHSLF              ;Current process
004367' 200 02 0 00 261316              MOVE    AC2,[LEVTAB,,CHNTAB]    ;Interrupt tables
004370' 104 00 0 00 000125              SIR%                            ;Set it up
004371' 205 02 0 00 760000              MOVSI   AC2,(ONCHNS)            ;Load channels to activate
004372' 104 00 0 00 000131              AIC%                            ;Activate channels
004373' 201 01 0 00 000005              MOVEI   AC1,AICLEN              ;Set up length of
004374' 202 01 0 00 004420'             MOVEM   AC1,.SQLEN+AICARG       ; the argument block
004375' 201 01 0 00 000020              MOVEI   AC1,.SSAIC              ;Function code
004376' 201 02 0 00 004420'             MOVEI   AC2,AICARG              ;Address of argument block
004377' 260 17 0 00 004605'             GO      DOSCS                   ;Execute the jsys
004400' 260 17 0 00 004633'              GO     ERRSCS                  ;If error go here
004401' 254 00 0 00 004413'             JRST    IFPIN5                  ;Exit

004402' 201 01 0 00 007307'     IFPIN1: MOVEI   AC1,PIVECT              ;Point to TOPS-10 Prog Int Vector
004403' 047 01 0 00 000135              PIINI.  AC1,                    ;Init that
004404' 254 00 0 00 261331               CRASH  (CII,Can't set PI system vector)
004405' 200 01 0 00 261337              MOVE    AC1,[PS.FAC!EXP .PCSCS,<SCOFFS,,0>,0]]
004406' 047 01 0 00 000136              PISYS.  AC1,                    ;Enable the channel
004407' 254 00 0 00 261355               CRASH  (CAI,<Can't activate interrupts for SCS events>)
004410' 200 01 0 00 261363              MOVE    AC1,[PS.FAC!EXP .PCTMR,<TIOFFS,,0>,0]]
004411' 047 01 0 00 000136              PISYS.  AC1,                    ;Enable the channel
004412' 254 00 0 00 261377               CRASH  (CST,<Can't set interrupts for PITMR>)

004413' 262 17 0 00 000004      IFPIN5: GET     AC4                     ;Restore the acs
004414' 262 17 0 00 000003              GET     AC3
004415' 262 17 0 00 000002              GET     AC2
004416' 262 17 0 00 000001              GET     AC1
004417' 263 17 0 00 000000              RTN                             ;Return

004420' 000000  000005          AICARG: 0,,AICLEN                       ;Words processed,, length of block
004421' 000000  000001                  .SIDGA,,DGMCHN                  ;Channel for datagram available
004422' 000001  000002                  .SIMSA,,MSGCHN                  ;Channel for message available
004423' 000002  000003                  .SIDMA,,DMACHN                  ;Channel for dma xfer complete
004424' 000003  000004                  .SIPAN,,OTHCHN                  ;Channel for connection & error events
                000005          AICLEN=.-AICARG                         ;Length of "AICARG" block
```

```
3460
3461                                    ;Disable the interrupt system.  The interrupts are remembered and will take
3462                                    ;effect when the interrupt system is reenabled.
3463
3464  004425' 261 17 0 00 000001    DISINT: PUT     AC1                     ;Save AC1
3465  004426' 336 00 0 00 000234'           SKIPN   MONTYP                  ;Skip if TOPS-orange
3466  004427' 254 00 0 00 004433'           JRST    DISIN1                  ;TOPS-blue
3467
3468  004430' 201 01 0 00 400000            MOVEI   AC1,.FHSLF              ;Current process
3469  004431' 104 00 0 00 000130            DIR%                            ;Disable the interrupt system
3470  004432' 254 00 0 00 004436'           JRST    DISIN2                  ;Restore AC and return
3471
3472  004433' 205 01 0 00 200000    DISIN1: MOVSI   AC1,(PS.FOF)            ;Turn off the interrupt system
3473  004434' 047 01 0 00 000136            PISYS.  AC1,                    ;Click
3474  004435' 254 00 0 00 261414             CRASH  (POF,Can't turn off PI system)
3475
3476  004436' 262 17 0 00 000001    DISIN2: GET     AC1                     ;Restore the AC
3477  004437' 263 17 0 00 000000            RTN                             ;And return
3478
3479                                    ;Reenable the interrupt system.
3480
3481  004440' 261 17 0 00 000001    ENAINT: PUT     AC1                     ;Save AC1
3482  004441' 336 00 0 00 000234'           SKIPN   MONTYP                  ;Skip if TOPS-20
3483  004442' 254 00 0 00 004446'           JRST    ENAIN1                  ;TOPS-blue
3484  004443' 201 01 0 00 400000            MOVEI   AC1,.FHSLF              ;Current process
3485  004444' 104 00 0 00 000126            EIR%                            ;Enable the interrupt system
3486  004445' 254 00 0 00 004451'           JRST    ENAIN2                  ;Exit
3487
3488  004446' 205 01 0 00 100000    ENAIN1: MOVSI   AC1,(PS.FON)            ;Load bit to enable the interrupts
3489  004447' 047 01 0 00 000136            PISYS.  AC1,                    ;Turn on the PI system
3490  004450' 254 00 0 00 261430             CRASH  (EON,Can't turn on PI system)
3491
3492  004451' 262 17 0 00 000001    ENAIN2: GET     AC1                     ;Restore AC1
3493  004452' 263 17 0 00 000000            RTN                             ;Return
```

```
3494
3495                                    ;Check to see if we are doing the wait in MAINLN. If so force a return
3496                                    ; to MAINLN code by changing LEVTAB (TOPS-20) or PIVECT (TOPS-10).  If not
3497                                    ;doing a wait AOS INTFLG.
3498
3499                                    ;Cwat for SCA traffic (TOPS-10 offset SCOFFS, TOPS-20 level 1)
3500
3501  004453' 261 17 0 00 000001    CWATSC: PUT     AC1                     ;Save an AC
3502  004454' 201 01 0 00 000000            MOVEI   AC1,SCOFFS              ;Load offset for SCS stuff
3503  004455' 260 17 0 00 004460'           GO      CWAT1                   ;Do the work
3504  004456' 262 17 0 00 000001            GET     AC1                     ;Restore the AC
3505  004457' 263 17 0 00 000000            RTN                             ;Return
3506
3507                                    ;CWAT for level 1 (TOPS-20) or for PI vector offset in AC1 (TOPS-10).
3508
3509  004460' 261 17 0 00 000000    CWAT1:  PUT     <AC0>                   ;Save an ac
3510  004461' 336 00 0 00 000221'           SKIPN   WAITF                   ;Are we in the 'WAIT" jsys?
3511  004462' 254 00 0 00 004467'           JRST    CWAT11                  ;No
3512  004463' 200 00 0 00 261433            MOVE    AC0,[1B5+MAINLN]        ;Load user mode bit + new PC
3513  004464' 202 00 1 00 007301'           MOVEM   AC0,@LEVTAB             ;Store it (orange)
3514  004465' 202 00 0 01 007310'           MOVEM   AC0,PIVECT+.PSVOP(AC1)  ;Store it (blue)
3515  004466' 402 00 0 00 000221'           SETZM   WAITF                   ;Clear the wait jsys flag
3516  004467' 350 00 0 00 000220'   CWAT11: AOS     INTFLG                  ;Increment the interrupt count
3517  004470' 262 17 0 00 000000            GET     <AC0>                   ;Restore ac
3518  004471' 263 17 0 00 000000            RTN                             ;Return to code that called this code
3519
3520                                    ;Enable capabilities for this job, only called at startup
3521
3522  004472' 336 00 0 00 000234'   ENABLE: SKIPN   MONTYP                  ;Is this an orange monitor?
3523  004473' 254 00 0 00 004503'           JRST    ENABL1                  ;Nope, blue monitor
3524  004474' 201 01 0 00 400000            MOVEI   AC1,.FHSLF              ;Process handle
3525  004475' 104 00 0 00 000150            RPCAP%                          ;Read capabilities
3526  004476' 320 16 0 00 004477'            ERJMP  .+1                     ;Ignore errors
3527  004477' 200 03 0 00 000002            MOVE    AC3,AC2                 ;Get capabilities to ac3
3528  004500' 104 00 0 00 000151            EPCAP%                          ;Now enable
3529  004501' 320 16 0 00 004502'            ERJMP  .+1                     ;Ignore errors
3530  004502' 263 17 0 00 000000            RTN                             ;Return
3531
3532  004503' 201 01 0 00 000017    ENABL1: MOVEI   AC1,LK.LLC!LK.LNP!LK.LNE!LK.LLS ;Lock low segment cached
3533  004504' 047 01 0 00 000060            LOCK    AC1,                    ;Or CALLI AC,60
3534  004505' 254 00 0 00 261444             CRASH  (CLJ,Cannot lock job)   ;Punt
3535  004506' 263 17 0 00 000000            RTN                             ;Return locked
```

```
                                        SUBTTL  Return Uptime/Time Of Day

                                ;Routine to return the number of milliseconds of uptime in AC1.
                                ;
                                ;Call: GO       UTIME

004507' 336 00 0 00 000234'     UTIME:  SKIPN   MONTYP          ;Skip if TOPS-20
004510' 254 00 0 00 004515'             JRST    UTIM1           ;TOPS-10
004511' 261 17 0 00 000002              PUT     AC2             ;Save AC2
004512' 104 00 0 00 000014              TIME%                   ;Get the uptime in ms
004513' 262 17 0 00 000002              GET     AC2             ;Restore AC2
004514' 263 17 0 00 000000              RTN                     ;Return

004515' 261 17 0 00 000002      UTIM1:  PUT     AC2             ;Save AC2
004516' 336 00 0 00 000235'             SKIPN   CYCL60          ;Computed 60HZ flag yet?
004517' 260 17 0 00 004527'             GO      GETTI2          ;No, get it now
004520' 200 01 0 00 261447              MOVE    AC1,[%CNSUP]    ;System uptime in jiffies %CNSUP
004521' 047 01 0 00 000041              GETTAB  AC1,            ;or CALLI AC1,41
004522' 400 01 0 00 000000               SETZ   AC1,            ;not likely it will fail
004523' 221 01 0 00 001750              IMULI   AC1,^D1000      ;convert to milli-jiffies
004524' 230 01 0 00 000235'             IDIV    AC1,CYCL60      ;Convert to milli-seconds
004525' 262 17 0 00 000002              GET     AC2             ;Restore the AC
004526' 263 17 0 00 000000              RTN                     ;Return

004527' 200 01 0 00 261450      GETTI2: MOVE    AC1,[%CNSTS]    ;Load %CNSTS system flags
004530' 047 01 0 00 000041              GETTAB  AC1,            ;or CALLI AC1,41
004531' 400 01 0 00 000000               SETZ   AC1,            ;should not execute this, assume 60
004532' 603 01 0 00 004000              TLNE    AC1,(1B6)       ;Is ST%CYC set for 50Hz?
004533' 334 01 0 00 261451              SKIPA   AC1,[^D50]      ;Yes, this must be Europe mon ami
004534' 201 01 0 00 000074              MOVEI   AC1,^D60        ;No, this is anywhere else
004535' 202 01 0 00 000235'             MOVEM   AC1,CYCL60      ;Set divisor
004536' 263 17 0 00 000000              RTN                     ; and return

                                ;This routine gets the date/time and returns it in ac1.

004537' 336 00 0 00 000234'     DATETM: SKIPN   MONTYP          ;Skip if TOPS-20
004540' 254 00 0 00 004543'             JRST    DATET1          ;TOPS-10
004541' 104 00 0 00 000227              GTAD%                   ;Get Time And Day
004542' 263 17 0 00 000000              RTN                     ;Return

004543' 200 01 0 00 261452      DATET1: MOVE    AC1,[%CNDTM]    ;Load gettab universal date-time
004544' 047 01 0 00 000041              GETTAB  AC1,            ;Or CALLI AC,41
004545' 400 01 0 00 000000               SETZ   AC1,            ;Error!
004546' 263 17 0 00 000000              RTN                     ;Return with date/time in AC1
```

```
                                        SUBTTL  Print Message Subroutines

                                ;Here to print a CRLF on terminal, smashed AC1

004547'                         PRINTC: HRROI   AC1,[ASCIZ/
004547' 561 01 0 00 261051      /]                                      ;Load crlf and fall thru

                                ;Here to print a message from AC1.
                                ;
                                ;Call:  HRROI   AC1,[ASCIZ/text/]
                                ;       GO      PRINTT                  ;Print message

004550' 332 00 0 00 000234'     PRINTT: SKIPE   MONTYP                  ;Skip if TOPS-10
004551' 104 00 0 00 000076              PSOUT%                          ;TOPS-20
004552' 336 00 0 00 000234'             SKIPN   MONTYP                  ;Skip if TOPS-20
004553' 051 03 0 01 000000              OUTSTR  (AC1)                   ;TOPS-10
004554' 263 17 0 00 000000              RTN                             ;Return

                                ;Here to print octal number in AC2.

004555' 336 00 0 00 000234'     PRINTO: SKIPN   MONTYP                  ;Skip if orange system
004556' 254 00 0 00 004573'             JRST    PRINT1                  ;Blue system
004557' 201 01 0 00 000101              MOVEI   AC1,.PRIOU              ;Output designator
004560' 201 03 0 00 000010              MOVEI   AC3,^D8                 ;Print in octal
004561' 104 00 0 00 000224              NOUT%                           ;Print function
004562' 320 16 0 00 004563'              ERJMP  .+1                     ;Ignore error
004563' 263 17 0 00 000000              RTN                             ;Return

                                ;Here to print decimal number in AC2.

004564' 336 00 0 00 000234'     PRINTD: SKIPN   MONTYP                  ;Skip if orange system
004565' 254 00 0 00 004574'             JRST    PRINT2                  ;Blue system
004566' 201 01 0 00 000101              MOVEI   AC1,.PRIOU              ;Output designator
004567' 201 03 0 00 000012              MOVEI   AC3,^D10                ;Print in decimal
004570' 104 00 0 00 000224              NOUT%                           ;Print function
004571' 320 16 0 00 004572'              ERJMP  .+1                     ;Ignore error
004572' 263 17 0 00 000000              RTN                             ;Return

004573' 334 01 0 00 261453      PRINT1: SKIPA   AC1,[DEC 8]             ;Here for octal output
004574' 201 01 0 00 000012      PRINT2: MOVEI   AC1,DEC 10              ;Here for decimal output
004575' 231 02 0 01 000000      PRINT3: IDIVI   AC2,(AC1)               ;Get a digit
004576' 271 03 0 00 000060              ADDI    AC3,"0"                 ;Convert to ASCII
004577' 261 17 0 00 000003              PUT     AC3                     ;Save the remainder
004600' 332 00 0 00 000002              SKIPE   AC2                     ;skip if all done
004601' 260 17 0 00 004575'             GO      PRINT3                  ;Not done, loop for all digits
004602' 262 17 0 00 000002              GET     AC2                     ;Get the remainder
004603' 051 01 0 00 000002              OUTCHR  AC2                     ;Print a digit
004604' 263 17 0 00 000000              RTN                             ;Return
```

```
                                        SUBTTL  Perform SCS JSYS/UUO

                                ;Here to do the appropriate SCS JSYS or UUO.
                                ;
                                ;Call:  MOVEI   AC1,function
                                ;       MOVEI   AC2,argblk
                                ;       GO      DOSCS
                                ;        error return
                                ;       normal return
                                ;
                                ;argblk: argument_block_length

004605' 124 01 0 00 C00240'     DOSCS:  DMOVEM  AC1,SCSFCN              ;Save SCS function code, arg block adr
004606' 250 01 0 17 000000              EXCH    AC1,(P)                 ;Get return PC, save S2
004607' 552 01 0 00 000236'             HRRZM   AC1,SCSCAL              ;Save calling PC
004610' 250 01 0 17 000000              EXCH    AC1,(P)                 ;Save return PC, get S2
004611' 336 00 0 00 000234'             SKIPN   MONTYP                  ;Skip if orange
004612' 254 00 0 00 004623'             JRST    DOSCS1                  ;TOPS-10 SCS JSYS

                                ;Here for TOPS-20.  Everything is setup properly, just do the JSYS.

004613' 104 00 0 00 000622              SCS%                            ;The JSYS
004614' 320 16 0 00 004617'             ERJMP   DOSCS2                  ;Error, non skip return
004615' 350 00 0 17 000000              AOS     (P)                     ;Skip return
004616' 263 17 0 00 000000              RTN                             ;Return+2

004617' 201 01 0 00 400000      DOSCS2: MOVEI   AC1,.FHSLF              ;Load this fork handle
004620' 104 00 0 00 000012              GETER%                          ;Get last error code
004621' 552 02 0 00 000237'             HRRZM   AC2,SCSERC              ;Store error code
004622' 263 17 0 00 000000              RTN                             ;Return+1

                                ;Here for TOPS-10.  The SCS UUO wants length,,functiocode in the first word
                                ; of the block were the SCS JSYS wants 0,,length.  Swap the arguments around.
                                ; Returns with error code in AC1 for decodeing by the ERRSCS routine.

004623' 504 01 0 02 000000      DOSCS1: HRL     AC1,(AC2)               ;Get length,,functioncode
004624' 202 01 0 02 000000              MOVEM   AC1,(AC2)               ;Save it back in the argument block
004625' 200 01 0 00 000002              MOVE    AC1,AC2                 ;Copy the address of the block
004626' 047 01 0 00 000213              SCS.    AC1,                    ;The UUO
004627' 304 00 0 00 000000               CAIA                           ;Error return
004630' 350 00 0 17 000000              AOS     (P)                     ;Give skip return
004631' 202 01 0 00 000237'             MOVEM   AC1,SCSERC              ;Save SCS error code
004632' 263 17 0 00 000000              RTN                             ;Return
```

```
3671
3672                                    ;SCS error routine.  This routine is called to tell us what error happened.
3673                                    ;SCSERC contains the error code.
3674
3675  004633' 336 00 0 00 000234'   ERRSCS: SKIPN   MONTYP                    ;Skip if TOPS-20
3676  004634' 254 00 0 00 004641'           JRST    ERRSC1                    ;TOPS-10
3677  004635' 561 01 0 00 261454            HRROI   AC1,[ASCIZ/? SCS JSYS error: /] ;Load initial label
3678  004636' 104 00 0 00 000076            PSOUT%                            ;Print it
3679  004637' 260 17 0 00 004676'           GO      ERSTRT                    ;Print the error code
3680  004640' 254 00 0 00 004654'           JRST    ERRSC4                    ;Continue below
3681
3682  004641' 051 03 0 00 261460    ERRSC1: OUTSTR  [ASCIZ/? SCS UUO error: (/] ;Print initial label
3683  004642' 200 02 0 00 000237'           MOVE    AC2,SCSERC                ;Copy error number
3684  004643' 260 17 0 00 004555'           GO      PRINTO                    ;Print that in octal
3685  004644' 051 03 0 00 261464            OUTSTR  [ASCIZ/) /]               ;End the error number
3686  004645' 200 01 0 00 000237'           MOVE    AC1,SCSERC                ;Restore error number
3687  004646' 205 02 0 00 777743            MOVSI   AC2,-ERRSCL               ;Load -ive length of table,,0
3688  004647' 554 03 0 02 004712'   ERRSC2: HLRZ    AC3,ERRSCT(AC2)           ;Load error code
3689  004650' 302 01 0 03 000000            CAIE    AC1,(AC3)                 ;Match?
3690  004651' 253 02 0 00 004647'           AOBJN   AC2,ERRSC2                ;Nope, loop
3691  004652' 550 02 0 02 004712'           HRRZ    AC2,ERRSCT(AC2)           ;Load address of ASCIZ text
3692  004653' 051 03 0 02 000000            OUTSTR  (AC2)                     ;Print that and fall thru
3693
3694  004654'                       ERRSC4: HRROI   AC1,[ASCIZ/
  Function /]                                                               ;Load more text
3695  004654' 561 01 0 00 261465
3696  004655' 260 17 0 00 004550'           GO      PRINTT                    ;Print that function code
3697  004656' 200 02 0 00 000240'           MOVE    AC2,SCSFCN                ;Load the function code
3698  004657' 260 17 0 00 004555'           GO      PRINTO                    ;Print that in octal please
3699  004660' 561 01 0 00 261470            HRROI   AC1,[ASCIZ/, argument block at /]
3700  004661' 260 17 0 00 004550'           GO      PRINTT                    ;Print label
3701  004662' 200 02 0 00 000241'           MOVE    AC2,SCSBLK                ;Load addr of error block
3702  004663' 260 17 0 00 004555'           GO      PRINTO                    ;Print addr in Octal
3703  004664' 561 01 0 00 261475            HRROI   AC1,[ASCIZ/, called from PC /] ;Print a label for the next bit
3704  004665' 260 17 0 00 004550'           GO      PRINTT                    ;Print that
3705  004666' 370 02 0 00 000236'           SOS     AC2,SCSCAL                ;Get pc-1
3706  004667' 260 17 0 00 004555'           GO      PRINTO                    ;Print octal number
3707  004670' 260 17 0 00 004547'           GO      PRINTC                    ;Print CRLF
3708
3709                                    ;Here to halt the program
3710
3711  004671' 332 00 0 00 000234'   HALTP:  SKIPE   MONTYP                    ;Skip if TOPS-10
3712  004672' 104 00 0 00 000170            HALTF%                            ;Stop orange
3713  004673' 336 00 0 00 000234'           SKIPN   MONTYP                    ;Skip if TOPS-20
3714  004674' 047 01 0 00 000012            EXIT    1,                        ;Stop blue
3715  004675' 254 00 0 00 000362'           JRST    START                     ;In case of a continue restart
```

```
3716
3717                                    ;Here to print the last JSYS error
3718
3719  004676' 201 01 0 00 000101    ERSTRT: MOVEI   AC1,.PRIOU              ;Destination designator
3720  004677' 200 02 0 00 000237'           MOVE    AC2,SCSERC              ;Load error code
3721  004700' 505 02 0 00 400000            HRLI    AC2,.FHSLF              ;Get process handle
3722  004701' 402 00 0 00 000003            SETZM   AC3                     ;Clr ac3
3723  004702' 104 00 0 00 000011            ERSTR%                          ;Tell about error
3724  004703' 320 17 0 00 004706'            ERCAL  PRINTA                  ;Undefined error number return
3725  004704' 320 17 0 00 004710'            ERCAL  PRINTB                  ;Erstr error
3726  004705' 263 17 0 00 000000            RTN                             ;Return
3727
3728  004706' 561 01 0 00 261501    PRINTA: HRROI   AC1,[ASCIZ/Undefined error number/] ;Undefined number
3729  004707' 254 00 0 00 004550'           JRST    PRINTT                  ;Print that
3730
3731  004710' 561 01 0 00 261506    PRINTB: HRROI   AC1,[ASCIZ/ERSTR error/] ;Other error type
3732  004711' 254 00 0 00 004550'           JRST    PRINTT                  ;Print that
3733
3734                                    ;Table of SCS UUO error codes
3735
3736  004712' 000000  261511        ERRSCT: XWD     SSNPV%,[ASCIZ/Not privileged/]
3737  004713' 000001  261514                XWD     SSIFC%,[ASCIZ/Illegal function code/]
3738  004714' 000002  261521                XWD     SSARG%,[ASCIZ/Bad argument list length/]
3739  004715' 000003  261526                XWD     SSACR%,[ASCIZ/Address check reading arguments/]
3740  004716' 000004  261535                XWD     SSACS%,[ASCIZ/Address check storing data/]
3741  004717' 000005  261543                XWD     SSCPN%,[ASCIZ/CPU number out of range/]
3742  004720' 000006  261550                XWD     SSNPC%,[ASCIZ/No CI port on specified CPU/]
3743  004721' 000007  261556                XWD     SSNNK%,[ASCIZ/CPU's node number not known/]
3744  004722' 000010  261564                XWD     SSINN%,[ASCIZ/Invalid CI node number/]
3745  004723' 000011  261571                XWD     SSNFC%,[ASCIZ/No free core/]
3746  004724' 000012  261574                XWD     SSVNO%,[ASCIZ/Virtual circuit not open/]
3747  004725' 000013  261601                XWD     SSICI%,[ASCIZ/Invalid connect id/]
3748  004726' 000014  261605                XWD     SSRQE%,[ASCIZ/Receive queue empty/]
3749  004727' 000015  261611                XWD     SSNBQ%,[ASCIZ/No buffer queued for packet reception/]
3750  004730' 000016  261621                XWD     SSRCF%,[ASCIZ/Reject connection failed/]
3751  004731' 000017  261626                XWD     SSDCF%,[ASCIZ/Disconnect connection failed/]
3752  004732' 000020  261634                XWD     SSNFB%,[ASCIZ/No free buffers to send packet/]
3753  004733' 000021  261643                XWD     SSQBF%,[ASCIZ/Queue buffers failed/]
3754  004734' 000022  261650                XWD     SSCBF%,[ASCIZ/Cancel buffers failed/]
3755  004735' 000023  261655                XWD     SSPSF%,[ASCIZ/Packet send failed/]
3756  004736' 000024  261661                XWD     SSDQE%,[ASCIZ/Data entry queue empty/]
3757  004737' 000025  261666                XWD     SSEQE%,[ASCIZ/Event queue empty/]
3758  004740' 000026  261672                XWD     SSCRB%,[ASCIZ/Can't remove buffer from database/]
3759  004741' 000027  261701                XWD     SSCUB%,[ASCIZ/Can't unmap buffer/]
3760  004742' 000030  261705                XWD     SSNSB%,[ASCIZ/No such buffer name/]
3761  004743' 000031  261711                XWD     SSTMS%,[ASCIZ/Too many buffer segment descriptors/]
3762  004744' 000032  261721                XWD     SSIDM%,[ASCIZ/Illegal data mode/]
3763  004745' 000033  261725                XWD     SSSCP%,[ASCIZ/Segment crosses page boundry/]
3764  004746' 000034  261733                XWD     SSSTL%,[ASCIZ/Segment too long (greater than 1 page)/]
3765  004747' 777777  261743                XWD     777777,[ASCIZ/Unknown error code/]
3766            000035              ERRSCL==.-ERRSCT-1                      ;Number of errors-1
```

```
                        SUBTTL  Send Unimplemented Opcode

                ;We come here if we get an unimplemented opcode.  This routine sets up the
                ;argument block for the scs% jsys monitor call.  The flag "VCGONE" is checked
                ;just before the jsys to see if the virtual circuit is still alive.
                ;
                ;Call:
                ;       Go      senimp                  ;call this routine
                ;       +1                              ;Always

004750' 261 17 0 00 000001      SENIMP: PUT     AC1                     ;Save the ac's
004751' 261 17 0 00 000002              PUT     AC2
004752' 261 17 0 00 000003              PUT     AC3
004753' 261 17 0 00 000004              PUT     AC4
004754' 201 02 0 00 000005              MOVEI   AC2,.LBSMG              ;Length of argument block
004755' 202 02 0 00 005010'             MOVEM   AC2,.SQLEN+NOTMSG       ;Store length of argument block
004756' 200 02 0 16 000000              MOVE    AC2,(CURSTS)            ;Get the "CID" of this connection &
004757' 202 02 0 00 005011'             MOVEM   AC2,.SQCID+NOTMSG       ; Store it in the argument block.
004760' 200 03 0 00 000276'             MOVE    AC3,CUREXE              ;Start of ctp
004761' 200 02 0 03 000003              MOVE    AC2,CTPPKT(AC3)         ; Request in
004762' 552 02 0 00 005012'             HRRZM   AC2,.SQAPT+NOTMSG       ; Argument block+2
004763' 200 03 0 00 000231'             MOVE    AC3,OPCODE              ;Use opcode for indexing
004764' 303 03 0 00 000020              CAILE   AC3,MAXOPC              ;If a valid request - skip
004765' 201 03 0 00 000021              MOVEI   AC3,MAXOPC+1            ;If not make value one more that max
004766' 200 02 0 03 007317'             MOVE    AC2,NMBYTS(AC3)         ;Get the number of
004767' 202 02 0 00 005013'             MOVEM   AC2,.SQLPT+NOTMSG       ; Bytes in the argument block
004770' 402 00 0 00 005014'             SETZM   .SQFLG+NOTMSG           ;Clear the flag word
004771' 201 01 0 00 000006              MOVEI   AC1,.SSSMG              ;Send a message
004772' 603 16 0 00 000020              TLNE    CURSTS,(DGM)            ;Did we receive this packet as a dgm?
004773' 201 01 0 00 000004              MOVEI   AC1,.SSSDG              ;Yes, send a datagram
004774' 201 02 0 00 005010'             MOVEI   AC2,NOTMSG              ;Address of argument block to ac2
004775' 603 16 0 00 000010              TLNE    CURSTS,(VCGONE)         ;Is this virtual circuit still around?
004776' 254 00 0 00 005001'             JRST    SENIMA                  ;No, take the +1 return
004777' 260 17 0 00 004605'             GO      DOSCS                   ;Do the SCS JSYS
005000' 260 17 0 00 004633'              GO     ERRSCS                  ;If error go here
005001' 332 00 0 00 000216'     SENIMA: TRMSG   (Send message or datagram - unimplemented opcode)
005002' 260 17 0 00 261761
005003' 262 17 0 00 000004              GET     AC4                     ;Restore the ac's
005004' 262 17 0 00 000003              GET     AC3
005005' 262 17 0 00 000002              GET     AC2
005006' 262 17 0 00 000001              GET     AC1
005007' 263 17 0 00 000000              RTN

                ;Send a message or datagram argument block

005010' 000000  000005          NOTMSG: 0,,.LBSMG                       ;Processed words,,length of block
005011' 000 00 0 00 000000              Z                               ;Connect id
005012' 000 00 0 00 000000              Z                               ;Address of message(rblk)
005013' 000 00 0 00 000000              Z                               ;Length of msg in 8 bit bytes
005014' 000 00 0 00 000000              Z                               ;Flags,,optional path spec
```

```
                                        SUBTTL  Send No Resource

                                ;This routine sends a "NO RESOURCE" status back to the controller.
                                ;We come here only during interrupt time.
                                ;Call:
                                ;       Move    ac1,[n]                 ;n=connect id
                                ;       Move    ac2,[m]                 ;m=buffer name(rblk)
                                ;       Move    ac3,[v]                 ;v=variable(-1=dgm 0=msg)
                                ;       Go      sennrs                  ;call this routine
                                ;       +1                              ;Always

005015' 261 17 0 00 000001      SENNRS: PUT     AC1                     ;Save the ac's
005016' 261 17 0 00 000002              PUT     AC2
005017' 261 17 0 00 000003              PUT     AC3
005020' 261 17 0 00 000004              PUT     AC4
005021' 201 04 0 00 000005              MOVEI   AC4,.LBSMG              ;Length of argument block
005022' 202 04 0 00 005050'             MOVEM   AC4,.SQLEN+NRSARG       ;Store length of argument block
005023' 202 01 0 00 005051'             MOVEM   AC1,.SQCID+NRSARG       ;Svore the cid
005024' 552 02 0 00 005052'             HRRZM   AC2,.SQAPT+NRSARG       ;Store the rblk address
005025' 200 04 0 00 000231'             MOVE    AC4,OPCODE              ;Use opcode for indexing
005026' 303 04 0 00 000020              CAILE   AC4,MAXOPC              ;If a valid request - skip
005027' 201 04 0 00 000021              MOVEI   AC4,MAXOPC+1            ;If not make value one more that max
005030' 200 02 0 04 007317'             MOVE    AC2,NMBYTS(AC4)         ;Get the number of
005031' 202 02 0 00 005053'             MOVEM   AC2,.SQLPT+NRSARG       ; Bytes in the argument block
005032' 402 00 0 00 005054'             SETZM   .SQFLG+NRSARG           ;Clear the flag word
005033' 201 01 0 00 000006              MOVEI   AC1,.SSSMG              ;Send a message function code
005034' 332 00 0 00 000003              SKIPE   AC3                     ;Was this received as a datagram?
005035' 201 01 0 00 000004              MOVEI   AC1,.SSSDG              ;Yes, then send back as a datagram
005036' 201 02 0 00 005050'             MOVEI   AC2,NRSARG              ;Address of argument block to ac2
005037' 260 17 0 00 004605'             GO      DOSCS                   ;Do the SCS JSYS
005040' 260 17 0 00 004633'              GO     ERRSCS                  ;If error go here

005041' 332 00 0 00 000216'     SENNRA: TRMSG   (Send message or datagram - no resource)
005042' 260 17 0 00 261777
005043' 262 17 0 00 000004              GET     AC4
005044' 262 17 0 00 000003              GET     AC3
005045' 262 17 0 00 000002              GET     AC2
005046' 262 17 0 00 000001              GET     AC1
005047' 263 17 0 00 000000              RTN

                                ;Send a message argument block (.SSSMG)

005050' 000000  000005          NRSARG: 0,,.LBSMG                       ;Processed words,,length of block
005051' 000 00 0 00 000000              Z                               ;Connect id
005052' 000 00 0 00 000000              Z                               ;Address of message(rblk)
005053' 000 00 0 00 000000              Z                               ;Length of msg in 8 bit bytes
005054' 000 00 0 00 000000              Z                               ;Flags,,optional path spec
```

```
                                        SUBTTL  Request Connection

                                ;This routine requests a connection with another node on the ci.
                                ;First we set up the argument block then execute the scs jsys.
                                ;3 Message buffers are queued with the connect function.
                                ;Call:
                                ;       Move    ac1,[n]                 ;n=remote node#
                                ;       Go      conect                  ;call this routine
                                ;       +1                              ;Ac1 has the 6 bit cid

005055' 261 17 0 00 000002      CONECT: PUT     AC2                     ;Save the ac's
005056' 261 17 0 00 000003              PUT     AC3
005057' 261 17 0 00 000004              PUT     AC4
005060' 201 02 0 00 000010              MOVEI   AC2,.LBCON              ;Length of argument block
005061' 202 02 0 00 005126'             MOVEM   AC2,.SQLEN+CONARG       ;Store length of argument block
005062' 200 02 0 00 000300'             MOVE    AC2,RSPNDR              ;Get the byte pointer of "CTP$RESPONDER"
005063' 202 02 0 00 005127'             MOVEM   AC2,.SQSPN+CONARG       ;Source process name
005064' 202 02 0 00 005130'             MOVEM   AC2,.SQDPN+CONARG       ;Destination process name
005065' 516 01 0 00 005131'             HRLZM   AC1,.SQSYS+CONARG       ;0,,Remote node# to remote node#,,0
005066' 260 17 0 00 003536'             GO      GETCID                  ;Get a connect id
005067' 542 01 0 00 005131'             HRRM    AC1,.SQSYS+CONARG       ;Store it
005070' 201 02 0 00 000311'             MOVEI   AC2,CONDAT              ;Get connection data word address
005071' 202 02 0 00 005132'             MOVEM   AC2,.SQCDT+CONARG       ;Store it in the argument block.

                                ;Queue up 3 buffers for this connect function.
                                ;In each buffer the 1st word is the address od the next buffer.
                                ;The last buffer has 0 to signify the end of the link.

005072' 260 17 0 00 003056'             GO      RHBLKV                  ;Get an available rhblk.
005073' 476 00 0 01 000000              SETOM   FLINK(AC1)              ;Make it "NOT AVAILABLE"
005074' 200 02 0 01 000003              MOVE    AC2,CTPPKT(AC1)         ;Get the corresponding rbx
005075' 202 02 0 00 005133'             MOVEM   AC2,.SQAMC+CONARG       ;Insert into the arg block
005076' 202 02 0 00 000003              MOVEM   AC2,AC3                 ;1St rbx to ac3
005077' 260 17 0 00 003056'             GO      RHBLKV                  ;Get an available rhblk.
005100' 476 00 0 01 000000              SETOM   FLINK(AC1)              ;Make it "NOT AVAILABLE"
005101' 200 02 0 01 000003              MOVE    AC2,CTPPKT(AC1)         ;Get the corresponding rbx
005102' 202 02 0 03 000000              MOVEM   AC2,(AC3)               ;Link this buffer
005103' 202 02 0 00 000003              MOVEM   AC2,AC3                 ;2Nd rbx to ac3
005104' 260 17 0 00 003056'             GO      RHBLKV                  ;Get an available rhblk.
005105' 476 00 0 01 000000              SETOM   FLINK(AC1)              ;Make it "NOT AVAILABLE"
005106' 200 02 0 01 000003              MOVE    AC2,CTPPKT(AC1)         ;Get the corresponding rbx
005107' 202 02 0 03 000000              MOVEM   AC2,(AC3)               ;Link this buffer
005110' 402 00 0 02 000000              SETZM   (AC2)                   ;Now stop the linking
```

```
3908
3909  005111' 402 00 0 00 005134'          SETZM   .SQADC+CONARG          ;Clear datagram buffer address
3910  005112' 402 00 0 00 005135'          SETZM   .SQRCI+CONARG          ;Clear previous returned cid
3911  005113' 201 01 0 00 000000           MOVEI   AC1,.SSCON             ;Connect function code
3912  005114' 201 02 0 00 005126'          MOVEI   AC2,CONARG             ;Address of argument block to ac2
3913  005115' 260 17 0 00 004605'          GO      DOSCS                  ;Do the jsys
3914  005116' 260 17 0 00 004633'           GO     ERRSCS                 ;If error go here
3915  005117' 550 01 0 00 005131'          HRRZ    AC1,.SQSYS+CONARG      ;Get the 6 bit cid in ac1
3916  005120' 332 00 0 00 000216'          TRMSG   (Connect)              ;Trace message
3917  005121' 260 17 0 00 262006
3918  005122' 262 17 0 00 000004           GET     AC4                    ;Restore the ac's
3919  005123' 262 17 0 00 000003           GET     AC3
3920  005124' 262 17 0 00 000002           GET     AC2
3921  005125' 263 17 0 00 000000           RTN                            ;Thats it
3922
3923                                      ;Connect (.SSCON) argument block
3924
3925  005126' 000000  000010              CONARG: 0,,.LBCON              ;Processed words,,length of block
3926  005127' 000 00 0 00 000000                  Z                      ;Byte ptr to source process name
3927  005130' 000 00 0 00 000000                  Z                      ;Byte ptr to destination process name
3928  005131' 000000  000000                      0,,0                   ;Node# of dest,,cid(6 bits)
3929  005132' 000000  000311'                     CONDAT                 ;Address to connection data
3930  005133' 000 00 0 00 000000                  Z                      ;Address of message buffer chain(rbx)
3931  005134' 000 00 0 00 000000                  Z                      ;Address of datagram buffer chain(rbx)
3932  005135' 000 00 0 00 000000                  Z                      ;Returned connect id(cid)
```

```
                                        SUBTTL  Listen for Controller

                                ;This routine will listen for a controller in any system.
                                ;First we set up the argument block then execute the scs jsys.
                                ;After the initialization code we come here only during interrupt time.
                                ;Call:
                                ;       Go      listnc                  ;call this routine
                                ;       +1                              ;Return here always

005136' 261 17 0 00 000001      LISTNC: PUT     AC1                     ;Save the ac's
005137' 261 17 0 00 000002              PUT     AC2
005140' 261 17 0 00 000003              PUT     AC3
005141' 261 17 0 00 000004              PUT     AC4
005142' 201 01 0 00 000005              MOVEI   AC1,.LBLIS              ;Get length of arg block
005143' 202 01 0 00 005172'             MOVEM   AC1,.SQLEN+LICARG       ;Store length of argument block
005144' 200 02 0 00 000300'             MOVE    AC2,RSPNDR              ;Get the addr of "CTP$RESPONDER"
005145' 202 02 0 00 005173'             MOVEM   AC2,.SQSPN+LICARG       ; Store it in the argument block.
005146' 200 02 0 00 000277'             MOVE    AC2,CNTRLR              ;Listen for a controller
005147' 202 02 0 00 005174'             MOVEM   AC2,.SQDPN+LICARG       ; Store it in the argument block.
005150' 515 02 0 00 777777              HRLZI   AC2,-1                  ;Listen for any system
005151' 202 02 0 00 005175'             MOVEM   AC2,.SQSYS+LICARG       ; Store it in the argument block.
005152' 260 17 0 00 003536'             GO      GETCID                  ;Get a connect id
005153' 542 01 0 00 005175'             HRRM    AC1,.SQSYS+LICARG       ;Store it
005154' 402 00 0 00 005176'             SETZM   .SQLCI+LICARG           ;Clear previous returned cid
005155' 201 01 0 00 000001              MOVEI   AC1,.SSLIS              ;Listen function code
005156' 201 02 0 00 005172'             MOVEI   AC2,LICARG              ;Address of argument block to ac2
005157' 260 17 0 00 004605'             GO      DOSCS                   ;Do the jsys
005160' 260 17 0 00 004633'              GO     ERRSCS                  ;If error go here
005161' 200 01 0 00 005176'             MOVE    AC1,.SQLCI+LICARG       ;Get the returned cid
005162' 202 01 0 00 000244'             MOVEM   AC1,CONIDC              ;Store it
005163' 332 00 0 00 000216'             TRMSG   (Listen for Controller)
005164' 260 17 0 00 262020
005165' 262 17 0 00 000004              GET     AC4                     ;Restore the ac's
005166' 262 17 0 00 000003              GET     AC3
005167' 262 17 0 00 000002              GET     AC2
005170' 262 17 0 00 000001              GET     AC1
005171' 263 17 0 00 000000              RTN                             ;Thats it

                                ;Listen (.SSLIS) for a controller argument block

005172' 000000  000005          LICARG: 0,,.LBLIS                       ;Processed words,,length of block
005173' 000 00 0 00 000000              Z                               ;Byte ptr to source process name
005174' 000 00 0 00 000000              Z                               ;Byte ptr to destination process name
005175' 000 00 0 00 000000              Z                               ;Node # of dest,,sysap field for cia
005176' 000 00 0 00 000000              Z                               ;Returned connect id(cid)
```

```
                                        SUBTTL  Listen for Responder

                                ;This routine will listen for a responder in any system.
                                ;First we set up the argument block then execute the scs jsys.
                                ;After the initialization code we come here only during interrupt time.
                                ;Call:
                                ;       Go      listnr                  ;call this routine
                                ;       +1                              ;Return here always

005177' 261 17 0 00 000001      LISTNR: PUT     AC1                     ;Save the ac's
005200' 261 17 0 00 000002              PUT     AC2
005201' 261 17 0 00 000003              PUT     AC3
005202' 261 17 0 00 000004              PUT     AC4
005203' 201 01 0 00 000005              MOVEI   AC1,.LBLIS              ;Get length
005204' 202 01 0 00 005232'             MOVEM   AC1,.SQLEN+LIRARG       ;Store length of argument block
005205' 200 02 0 00 000300'             MOVE    AC2,RSPNDR              ;Get the byte ptr of "CTP$RESPONDER"
005206' 202 02 0 00 005233'             MOVEM   AC2,.SQSPN+LIRARG       ; Store it for src process name
005207' 202 02 0 00 005234'             MOVEM   AC2,.SQDPN+LIRARG       ; Store it for dst process name
005210' 515 02 0 00 777777              HRLZI   AC2,-1                  ;Listen for any system
005211' 202 02 0 00 005235'             MOVEM   AC2,.SQSYS+LIRARG       ; Store it in the argument block.
005212' 260 17 0 00 003536'             GO      GETCID                  ;Get a connect id
005213' 542 01 0 00 005235'             HRRM    AC1,.SQSYS+LIRARG       ; Store it in the argument block.
005214' 402 00 0 00 005236'             SETZM   .SQLCI+LIRARG           ;Clear previous returned cid
005215' 201 01 0 00 000001              MOVEI   AC1,.SSLIS              ;Listen function code
005216' 201 02 0 00 005232'             MOVEI   AC2,LIRARG              ;Address of argument block to ac2
005217' 260 17 0 00 004605'             GO      DOSCS                   ;Do the jsys
005220' 260 17 0 00 004633'              GO     ERRSCS                  ;If error go here
005221' 200 01 0 00 005236'             MOVE    AC1,.SQLCI+LIRARG       ;Get the returned cid
005222' 202 01 0 00 000245'             MOVEM   AC1,CONIDR              ;Store it
005223' 332 00 0 00 000216'             TRMSG   (Listen for Responder)
005224' 260 17 0 00 262032

005225' 262 17 0 00 000004                      GET     AC4             ;Restore the ac's
005226' 262 17 0 00 000003                      GET     AC3
005227' 262 17 0 00 000002                      GET     AC2
005230' 262 17 0 00 000001                      GET     AC1
005231' 263 17 0 00 000000                      RTN                     ;Thats it

                                ;Listen(.Sslis) for a responder argument block

005232' 000000  000005          LIRARG: 0,,.LBLIS                       ;Processed words,,length of block
005233' 000 00 0 00 000000              Z                               ;Byte ptr to source process name
005234' 000 00 0 00 000000              Z                               ;Byte ptr to destination process name
005235' 000 00 0 00 000000              Z                               ;Node # of dest,,sysap field for cid
005236' 000 00 0 00 000000              Z                               ;Returned connect id(cid)
```

```
                                        SUBTTL  Accept Connection

                                ;This routine tells a remote node that we accept his request for a connection.
                                ;First we set up the argument block then execute the scs jsys.
                                ;We come here only during interrupt time.
                                ;Call:
                                ;       Move    ac1,[n]                 ;n=connect id
                                ;       Go      accept                  ;call this routine
                                ;       +1                              ;Return here always

005237' 261 17 0 00 000001      ACCEPT: PUT     AC1                     ;Save the ac's
005240' 261 17 0 00 000002              PUT     AC2
005241' 261 17 0 00 000003              PUT     AC3
005242' 261 17 0 00 000004              PUT     AC4
005243' 201 02 0 00 000003              MOVEI   AC2,.LBACC              ;Load length
005244' 202 02 0 00 005263'             MOVEM   AC2,.SQLEN+ACCARG       ;Store length of argument block
005245' 202 01 0 00 005264'             MOVEM   AC1,.SQCID+ACCARG       ;Store the connect id
005246' 201 02 0 00 000311'             MOVEI   AC2,CONDAT              ;Get connection data word address
005247' 202 02 0 00 005265'             MOVEM   AC2,.SQCDA+ACCARG       ; Store it in the argument block.
005250' 201 01 0 00 000023              MOVEI   AC1,.SSACC              ;Accept function code
005251' 201 02 0 00 005263'             MOVEI   AC2,ACCARG              ;Address of argument block to ac2
005252' 260 17 0 00 004605'             GO      DOSCS                   ;Do the jsys
005253' 260 17 0 00 004633'              GO     ERRSCS                  ;If error go here
005254' 332 00 0 00 000216'             TRMSG   (Accept Connection)
005255' 260 17 0 00 262043

005256' 262 17 0 00 000004              GET     AC4                     ;Restore the ac's
005257' 262 17 0 00 000003              GET     AC3
005260' 262 17 0 00 000002              GET     AC2
005261' 262 17 0 00 000001              GET     AC1
005262' 263 17 0 00 000000              RTN                             ;Thats it

                                ;Accept(.Ssacc) argument block

005263' 000000  000003          ACCARG: 0,,.LBACC                       ;Processed words,,length of block
005264' 000 00 0 00 000000              Z                               ;Connect id
005265' 000000  000311'                 CONDAT                          ;Ptr to connection data
```

```
                                        SUBTTL  Reject Connection

                                ;This routine tells a remote node that we reject his request for a connection.
                                ;First we set up the argument block then execute the scs jsys.
                                ;We come here only during interrupt time.
                                ;Call:
                                ;       Move    ac1,[n]                 ;n=connect id
                                ;       Move    ac2,[m]                 ;m=reject reason
                                ;       Go      reject                  ;call this routine
                                ;       +1                              ;Return here always

005266' 261 17 0 00 000001      REJECT: PUT     AC1                     ;Save the ac's
005267' 261 17 0 00 000002              PUT     AC2
005270' 261 17 0 00 000003              PUT     AC3
005271' 261 17 0 00 000004              PUT     AC4
005272' 201 04 0 00 000003              MOVEI   AC4,.LBREJ              ;Load length
005273' 202 04 0 00 005311'             MOVEM   AC4,.SQLEN+REJARG       ;Store length of argument block
005274' 202 01 0 00 005312'             MOVEM   AC1,.SQCID+REJARG       ;Store the connect id
005275' 202 02 0 00 005313'             MOVEM   AC2,.SQREJ+REJARG       ;Store the reject reason
005276' 201 01 0 00 000002              MOVEI   AC1,.SSREJ              ;Reject function code
005277' 201 02 0 00 005311'             MOVEI   AC2,REJARG              ;Address of argument block to ac2
005300' 260 17 0 00 004605'             GO      DOSCS                   ;The jsys
005301' 260 17 0 00 004633'              GO     ERRSCS                  ;If error go here

005302' 332 00 0 00 000216'             TRMSG   (Reject Connection)
005303' 260 17 0 00 262054

005304' 262 17 0 00 000004              GET     AC4                     ;Restore the ac's
005305' 262 17 0 00 000003              GET     AC3
005306' 262 17 0 00 000002              GET     AC2
005307' 262 17 0 00 000001              GET     AC1
005310' 263 17 0 00 000000              RTN                             ;Thats it

                                ;Reject (.SSREJ) argument block

005311' 000000  000003          REJARG: 0,,.LBREJ                       ;Processed words,,length of block
005312' 000 00 0 00 000000              Z                               ;Connect id
005313' 000 00 0 00 000000              Z                               ;Reject reason
```

```
                                        SUBTTL  Close Connection

                                ;This routine closes a connection.
                                ;First we set up the argument block then execute the scs jsys.
                                ;Call:
                                ;       Move    ac1,[n]                 ;n=connect id
                                ;       Move    ac2,[m]                 ;m=disconnect reason
                                ;       Go      discon                  ;call this routine
                                ;       +1                              ;Return here always

005314' 261 17 0 00 000001      DISCON: PUT     AC1                     ;Save the ac's
005315' 261 17 0 00 000002              PUT     AC2
005316' 261 17 0 00 000003              PUT     AC3
005317' 261 17 0 00 000004              PUT     AC4
005320' 201 04 0 00 000003              MOVEI   AC4,.LBDIS              ;Load length
005321' 202 04 0 00 005337'             MOVEM   AC4,.SQLEN+DISARG       ;Store length of argument block
005322' 202 01 0 00 005340'             MOVEM   AC1,.SQCID+DISARG       ;Store the connect id
005323' 202 02 0 00 005341'             MOVEM   AC2,.SQDIS+DISARG       ;Store the disconnect reason
005324' 201 01 0 00 000003              MOVEI   AC1,.SSDIS              ;Disconnect function code
005325' 201 02 0 00 005337'             MOVEI   AC2,DISARG              ;Address of argument block to ac2
005326' 260 17 0 00 004605'              GO     DOSCS                           ;The jsys
005327' 260 17 0 00 004633'              GO     ERRSCS                  ;If error go here

005330' 332 00 0 00 000216'             TRMSG   (Disconnect)
005331' 260 17 0 00 262064

005332' 262 17 0 00 000004              GET     AC4                     ;Restore the ac's
005333' 262 17 0 00 000003              GET     AC3
005334' 262 17 0 00 000002              GET     AC2
005335' 262 17 0 00 000001              GET     AC1
005336' 260 17 0 00 000000              RTN                             ;Thats it

                                ;Disconnect (.SSDIS) argument block

005337' 000000  000003          DISARG: 0,,.LBDIS                       ;Processed words,,length of block
005340' 000 00 0 00 000000              Z                               ;Connect id
005341' 000 00 0 00 000000              Z                               ;Disconnect reason
```

```
                                        SUBTTL  Queue Message Buffer

                                ;This routine first sets up the argument block then executes an scs
                                ;  'QUEUE A BUFFER FOR MESSAGE RECEPTION'.
                                ;The interrupt system must be shut of before entering this routine and must
                                ;Enabled after execution of this subroutine.
                                ;
                                ;CALL:
                                ;       Move    ac1,[n]                 ;n=connect id
                                ;       Move    ac2,[m]                 ;m=buffer name(rblk)
                                ;       Go      qbmsg                   ;call this routine
                                ;       +1                              ;Return here always

005342' 261 17 0 00 000001      QBMSG:  PUT     AC1                     ;Save the ac's
005343' 261 17 0 00 000002              PUT     AC2
005344' 261 17 0 00 000003              PUT     AC3
005345' 261 17 0 00 000004              PUT     AC4
005346' 201 04 0 00 000003              MOVEI   AC4,.LBQRM              ;Load length
005347' 202 04 0 00 005365'             MOVEM   AC4,.SQLEN+QBMARG       ;Store length of argument block
005350' 202 01 0 00 005366'             MOVEM   AC1,.SQCID+QBMARG       ;Store connect id
005351' 202 02 0 00 005367'             MOVEM   AC2,.SQAFB+QBMARG       ;Store the buffer name
005352' 201 01 0 00 000007              MOVEI   AC1,.SSQRM              ;Queue a buffer for datagrams func code
005353' 201 02 0 00 005365'             MOVEI   AC2,QBMARG              ;Address of argument block to ac2
005354' 260 17 0 00 004605'             GO      DOSCS                   ;Execute the jsys
005355' 260 17 0 00 004633'              GO     ERRSCS                  ;If error go here

005356' 332 00 0 00 000216'             TRMSG   (Queue Message Buffer)
005357' 260 17 0 00 262076
005360' 262 17 0 00 000004              GET     AC4                     ;Restore the ac's
005361' 262 17 0 00 000003              GET     AC3
005362' 262 17 0 00 000002              GET     AC2
005363' 262 17 0 00 000001              GET     AC1
005364' 263 17 0 00 000000              RTN                             ;Thats it

                                ;queue a buffer for message reception(.Ssqrm) argument block

005365' 000000  000003          QBMARG: 0,,.LBQRM                       ;Processed words,,length of block
005366' 000 00 0 00 000000              Z                               ;Connect id
005367' 000 00 0 00 000000              Z                               ;Address of 1st buffer in chain
```

```
                                        SUBTTL  Queue Datagram Buffer

                                ;This routine first sets up the argument block then executes an scs
                                ; "QUEUE A BUFFER FOR DATAGRAM RECEPTION".
                                ;The interrupt system must be shut of before entering this routine and must
                                ;Enabled after execution of this subroutine.
                                ;Call:
                                ;       Move    ac1,[n]                 ;n=connect id
                                ;       Move    ac2,[m]                 ;m=buffer name(rblk)
                                ;       Go      qbdgm                   ;call this routine
                                ;       +1                              ;Return here always

005370' 261 17 0 00 000001      QBDGM:  PUT     AC1                     ;Save the ac's
005371' 261 17 0 00 000002              PUT     AC2
005372' 261 17 0 00 000003              PUT     AC3
005373' 261 17 0 00 000004              PUT     AC4
005374' 201 04 0 00 000003              MOVEI   AC4,.LBQRD              ;Load length
005375' 202 04 0 00 005413'             MOVEM   AC4,.SQLEN+QBDARG       ;Store length of argument block
005376' 202 01 0 00 005414'             MOVEM   AC1,.SQCID+QBDARG       ;Store connect id
005377' 202 02 0 00 005415'             MOVEM   AC2,.SQAFB+QBDARG       ;Store the buffer name
005400' 201 01 0 00 000005              MOVEI   AC1,.SSQRD              ;Queue a buffer fcr datagrams func code
005401' 201 02 0 00 005413'             MOVEI   AC2,QBDARG              ;Address of argument block to ac2
005402' 260 17 0 00 004605'             GO      DOSCS                   ;Execute the jsys
005403' 260 17 0 00 004633'              GO     ERRSCS                  ;If error go here

005404' 332 00 0 00 000216'             TRMSG   (Queue Datagram Buffer)
005405' 260 17 0 00 262110

005406' 262 17 0 00 000004              GET     AC4                     ;Restore the ac's
005407' 262 17 0 00 000003              GET     AC3
005410' 262 17 0 00 000002              GET     AC2
005411' 262 17 0 00 000001              GET     AC1
005412' 263 17 0 00 000000              RTN                             ;Thats it

                                ,Queue a buffer for datagram reception(.Ssqrd) argument block

005413' 000000  000003          QBDARG: 0,,.LBQRD                       ;Processed words,,length of block
005414' 000 00 0 00 000000              Z                               ;Connect id
005415' 000 00 0 00 000000              Z                               ;Address of 1st buffer in chain
```

```
                                        SUBTTL  Recieve Message

                                ;This routine returns message text taken off the wire for your connection.
                                ;First we set up the argument block then execute the scs jsys.
                                ;We come here only during interrupt time.
                                ;Call:
                                ;       Go      rcvmsg                  ;call this routine
                                ;       +1                              ;Return here always

005416' 261 17 0 00 000001      RCVMSG: PUT     AC1                     ;Save the ac's
005417' 261 17 0 00 000002              PUT     AC2
005420' 261 17 0 00 000003              PUT     AC3
005421' 261 17 0 00 000004              PUT     AC4
005422' 201 01 0 00 000005              MOVEI   AC1,.LBRMG
005423' 202 01 0 00 005445'             MOVEM   AC1,.SQLEN+RCMARG       ;Store length of argument block
005424' 476 00 0 00 005446'             SETOM   .SQCID+RCMARG           ;-1 = Get any msg found for this fork
005425' 402 00 0 00 005447'             SETZM   .SQARB+RCMARG           ;Clear previous address buffer
005426' 402 00 0 00 005450'             SETZM   .SQDFL+RCMARG           ;Clear previous flags
005427' 402 00 0 00 005451'             SETZM   .SQLRP+RCMARG           ;Clear previous length of rtnd msg
005430' 201 01 0 00 000013              MOVEI   AC1,.SSRMG              ;Receive msg function code
005431' 201 02 0 00 005445'             MOVEI   AC2,RCMARG              ;Address of argument block to ac2
005432' 260 17 0 00 004605'             GO      DOSCS                   ;Execute the jsys
005433' 254 00 0 00 005435'              JRST   RCVMSE                  ;If error go here(no more messages)
005434' 334 00 0 00 000000              SKIPA
005435' 402 00 0 00 005447'     RCVMSE: SETZM   .SQARB+RCMARG           ;Flag the error

005436' 332 00 0 00 000216'             TRMSG   (Recieve Message)
005437' 260 17 0 00 262121

005440' 262 17 0 00 000004              GET     AC4                     ;Restore the ac's
005441' 262 17 0 00 000003              GET     AC3
005442' 262 17 0 00 000002              GET     AC2
005443' 262 17 0 00 000001              GET     AC1
005444' 263 17 0 00 000000              RTN                             ;Thats it

                                ;Receive a message(.Ssrmg) argument block

005445' 000000  000005          RCMARG: 0,,.LBRMG                       ;Processed words,,length of block
005446' 000 00 0 00 000000              Z                               ;Returned - cid
005447' 000 00 0 00 000000              Z                               ;Returned - address of buffer
005450' 000 00 0 00 000000              Z                               ;Returned - flags,,remote node#
005451' 000 00 0 00 000000              Z                               ;Length of returned message
```

```
                                SUBTTL  Recieve Datagram

                        ;This routine returns datagram data taken off the wire for your connection.
                        ;First we set up the argument block then execute the scs jsys.
                        ;We come here only during interrupt time.
                        ;Call:
                        ;       Go      rcvdgm                  ;call this routine
                        ;       +1                              ;Return here always

005452' 261 17 0 00 000001      RCVDGM: PUT     AC1             ;Save the ac's
005453' 261 17 0 00 000002              PUT     AC2
005454' 261 17 0 00 000003              PUT     AC3
005455' 261 17 0 00 000004              PUT     AC4
005456' 201 01 0 00 000005              MOVEI   AC1,.LBRDG      ;Load length
005457' 202 01 0 00 005501'             MOVEM   AC1,.SQLEN+RCDARG       ;Store length of argument block
005460' 476 00 0 00 005502'             SETOM   .SQCID+RCDARG   ;-1 = Get any msg found for this fork
005461' 402 00 0 00 005503'             SETZM   .SQARB+RCDARG   ;Clear previous address buffer
005462' 402 00 0 00 005504'             SETZM   .SQDFL+RCDARG   ;Clear previous flags
005463' 402 00 0 00 005505'             SETZM   .SQLRP+RCDARG   ;Clear previous length of rtnd msg
005464' 201 01 0 00 000022              MOVEI   AC1,.SSRDG      ;Receive dgm function code
005465' 201 02 0 00 005501'             MOVEI   AC2,RCDARG      ;Address of argument block to ac2
005466' 260 17 0 00 004605'             GO      DOSCS           ;Execute the jsys
005467' 254 00 0 00 005471'              JRST   RCVDGE          ;If error go here(no more datagrams)
005470' 334 00 0 00 000000              SKIPA
005471' 402 00 0 00 005503'     RCVDGE: SETZM   .SQARB+RCDARG   ;Flag the error

005472' 332 00 0 00 000216'             TRMSG   (Recieve Datagram)
005473' 260 17 0 00 262132

005474' 262 17 0 00 000004              GET     AC4             ;Restore the ac's
005475' 262 17 0 00 000003              GET     AC3
005476' 262 17 0 00 000002              GET     AC2
005477' 262 17 0 00 000001              GET     AC1
005500' 263 17 0 00 000000              RTN                     ;Thats it

                        ;Receive a message(.Ssrdg) argument block

005501' 000000  000005          RCDARG: 0,,.LBRDG               ;Processed words,,length of block
005502' 000 00 0 00 000000              Z                       ;Returned - cid
005503' 000 00 0 00 000000              Z                       ;Returned - address of buffer
005504' 000 00 0 00 000000              Z                       ;Returned - flags,,remote node#
005505' 000 00 0 00 000000              Z                       ;Length of returned message
```

```
                                        SUBTTL  Send Message or Datagram

                                ;This routine sends either a "DATAGRAM" or "MESSAGE".
                                ;Set up the argument block for the scs% jsys monitor call.
                                ;The flag "VCGONE" is checked just before the jsys to see
                                ;If the virtual circuit is still alive.
                                ;Call:
                                ;       Move    1,[n]                   ;n=connect id
                                ;       Move    2,[m]                   ;m=address of ctp packet(rblk)
                                ;       Move    3,[x]                   ;x=length in 8 bit bytes
                                ;       Go      sendmd                  ;call this routine
                                ;       +1                              ;Error return
                                ;       +2                              ;Successful return

005506' 261 17 0 00 000001      SENDMD: PUT     AC1                     ;Save the ac's
005507' 261 17 0 00 000002              PUT     AC2
005510' 261 17 0 00 000003              PUT     AC3
005511' 261 17 0 00 000004              PUT     AC4
005512' 201 00 0 00 000005              MOVEI   AC0,.LBSMG              ;Length
005513' 202 00 0 00 005550'             MOVEM   AC0,.SQLEN+MSGDGM       ;Store length of argument block
005514' 202 01 0 00 005551'             MOVEM   AC1,.SQCID+MSGDGM       ;Store the cid in the argument block.
005515' 552 02 0 00 005552'             HRRZM   AC2,.SQAPT+MSGDGM       ;Store the start address
005516' 202 03 0 00 005553'             MOVEM   AC3,.SQLPT+MSGDGM       ;Store length of datagram
005517' 402 00 0 00 005554'             SETZM   .SQFLG+MSGDGM           ;Clear the flags

005520' 201 01 0 00 000006              MOVEI   AC1,.SSSMG              ;Send a message
005521' 603 16 0 00 000020              TLNE    CURSTS,(DGM)            ;Did we receive this packet as a dgm?
005522' 201 01 0 00 000004              MOVEI   AC1,.SSSDG              ;Yes, send a datagram

005523' 201 02 0 00 005550'             MOVEI   AC2,MSGDGM              ;Address of argument block to ac2
005524' 603 16 0 00 000010              TLNE    CURSTS,(VCGONE)         ;Is this virtual circuit still around?
005525' 254 00 0 00 005543'             JRST    SENDMC                  ;No, take the +1 return
005526' 200 03 0 00 000231'             MOVE    AC3,OPCODE              ;Get the opcode
005527' 302 03 0 00 000005              CAIE    AC3,5                   ;Are we doing the msg
005530' 306 03 0 00 000004              CAIN    AC3,4                   ; Or dgm test?
005531' 336 00 0 00 000222'             SKIPN   LASTF                   ;Yes, is this the last xfer?
005532' 254 00 0 00 005536'             JRST    SENDMA                  ;No
```

```
                                repeat 0,<                              ;[0.3] Confused..
                                        TLN     CURSTS,(DGMSG)          ;Did we receive this as a msg?
                                        MOVEI   AC13,[DGM+DGMSG]        ;Get the dgm and dgmsg bits
                                        MOVE    AC3,CUREXE              ;Get the rhblk
                                        TDZ     AC13,STATUS(AC3)        ;Yes, clr dgm and dgmsg bits
                                >;end of repeat 0
005533' 661 16 0 00 000002              TLO     CURSTS,(DONE)           ;Yes, set the done bit
005534' 200 03 0 00 000276'             MOVE    AC3,CUREXE              ;Get the rhblk
005535' 202 16 0 03 000002              MOVEM   CURSTS,STATUS(AC3)      ;Store the status word
005536' 260 17 0 00 004605'     SENDMA: GO      DOSCS                   ;Execute the jsys
005537' 260 17 0 00 004633'              GO     ERRSCS                  ;If error go here

005540' 332 00 0 00 000216'             TRMSG   (Send Datagram or Message)
005541' 260 17 0 00 262145

005542' 350 00 0 17 777774              AOS     -4(P)                   ;Make it a +2 return
005543' 262 17 0 00 000004      SENDMC: GET     AC4                     ;Restore the ac's
005544' 262 17 0 00 000003              GET     AC3
005545' 262 17 0 00 000002              GET     AC2
005546' 262 17 0 00 000001              GET     AC1
005547' 263 17 0 00 000000              RTN                             ;Thats it

                                ;Send a message or datagram argument block

00555C' 000000  000005          MSGDGM: 0,,.LBSMG                       ;Processed words,,length of block
005551' 000 00 0 00 000000              Z                               ;Connect id
005552' 000 00 0 00 000000              Z                               ;Address of message(rblk)
005553' 000 00 0 00 000000              Z                               ;Length of msg in 8 bit bytes
005554' 000 00 0 00 000000              Z                               ;Flags,,optional path spec
```

```
                                        SUBTTL  Connection State Poll

                                ;This routine returns detailed information about a connection.
                                ;First we set up the argument block then execute the scs jsys.
                                ;Call:
                                ;       Move    ac1,[n]                 ;n=connect id
                                ;       Go      stpoll                  ;call this routine
                                ;       +1                              ;Return here always

005555' 261 17 0 00 000001      STPOLL: PUT     AC1                     ;Save the ac's
005556' 261 17 0 00 000002              PUT     AC2
005557' 261 17 0 00 000003              PUT     AC3
005560' 261 17 0 00 000004              PUT     AC4
005561' 201 04 0 00 000007              MOVEI   AC4,.LBCSP
005562' 202 04 0 00 005626'             MOVEM   AC4,.SQLEN+STPARG       ;Store length of argument block
005563' 202 01 0 00 005627'             MOVEM   AC1,.SQCID+STPARG       ;Store the connect id
005564' 200 02 0 00 262152              MOVE    AC2,[POINT 7,DUM1]      ;Free space block
005565' 202 02 0 00 005632'             MOVEM   AC2,.SQBDN+STPARG       ;Store the destination process name
005566' 201 01 0 00 000010              MOVEI   AC1,.SSCSP              ;Connect state poll function code
005567' 201 02 0 00 005626'             MOVEI   AC2,STPARG              ;Address of argument block to ac2
005570' 260 17 0 00 004605'             GO      DOSCS                   ;Execute the jsys
005571' 260 17 0 00 004633'              GO     ERRSCS                  ;If error go here

005572' 332 00 0 00 000216'             TRMSG   (Connection State Poll)
005573' 260 17 0 00 262160
```

```
                                ;Test that the returned destination process name is correct ctp$controller

005574' 200 01 0 00 000301'             MOVE    AC1,CNTLR               ;Get correct
005575' 312 01 0 00 005620'             CAME    AC1,DUM1                ;Is returned process name the same ?
005576' 254 00 0 00 005606'             JRST    STPOL1                  ; No, error
005577' 200 01 0 00 000302'             MOVE    AC1,CNTLR+1             ;Get correct
005600' 312 01 0 00 005621'             CAME    AC1,DUM1+1              ;Is returned process name the same ?
005601' 254 00 0 00 005606'             JRST    STPOL1                  ; No, error
005602' 200 01 0 00 000303'             MOVE    AC1,CNTLR+2             ;Get correct
005603' 312 01 0 00 005622'             CAME    AC1,DUM1+2              ;Is returned process name the same ?
005604' 254 00 0 00 005606'             JRST    STPOL1                  ; No, error
005605' 254 00 0 00 005613'             JRST    STPOL2                  ; It's correct

005606' 561 01 0 00 262165      STPOL1: HRROI   AC1,[ASCIZ/? STPOLL - incorrect process name /]
005607' 260 17 0 00 004550'             GO      PRINTT                  ;Print that
005610' 561 01 0 00 005620'             HRROI   AC1,DUM1
005611' 260 17 0 00 004550'             GO      PRINTT                  ;Print that
005612' 260 17 0 00 004547'             GO      PRINTC                  ;Print CRLF

005613' 262 17 0 00 000004      STPOL2: GET     AC4                     ;Restore the ac's
005614' 262 17 0 00 000003              GET     AC3
005615' 262 17 0 00 000002              GET     AC2
005616' 262 17 0 00 000001              GET     AC1
005617' 263 17 0 00 000000              RTN                             ;Thats it

005620'                         DUM1:   BLOCK   6                       ;Storage for destination process name

                                ;Connect state poll(.Sscsp) argument block

005626' 000000  000007          STPARG: 0,,.LBCSP                       ;Processed words,,length of block
005627' 000 00 0 00 000000              Z                               ;Connect id
005630' 000 00 0 00 000000              Z                               ;Connection state
005631' 000 00 0 00 000000              Z                               ;Destination connect id
005632' 000 00 0 00 000000              Z                               ;Pointer to destination process name
005633' 000 00 0 00 000000              Z                               ;Remote node# of destination
005634' 000 00 0 00 000000              Z                               ;Source connect reasons,,dest disconnect reasons
```

```
                                        SUBTTL  Return Local Node Number

                                ;This routine returns the local node number.
                                ;First we set up the argument block then execute the scs jsys.
                                ;Call:
                                ;       Go      localn                  ;call this routine
                                ;       +1                              ;Return here always
                                ;
005635' 261 17 0 00 000001      LOCALN: PUT     AC1                     ;Save the ac's
005636' 261 17 0 00 000002              PUT     AC2
005637' 261 17 0 00 000003              PUT     AC3
005640' 261 17 0 00 000004              PUT     AC4
005641' 201 04 0 00 000002              MOVEI   AC4,.LBGLN              ;Get the length of arg block
005642' 202 04 0 00 005657'             MOVEM   AC4,.SQLEN+LOCARG       ;Store length of argument block
005643' 402 00 0 00 005660'             SETZM   .SQLNN+LOCARG           ;Clear the node# word
005644' 201 01 0 00 000030              MOVEI   AC1,.SSGLN              ;Return local node# function code
005645' 201 02 0 00 005657'             MOVEI   AC2,LOCARG              ;Address of argument block to ac2
005646' 260 17 0 00 004605'             GO      DOSCS                   ;Execute the jsys
005647' 260 17 0 00 004633'              GO     ERRSCS                  ;if error go here

005650' 332 00 0 00 000216'             TRMSG   (Return Local Node Number)
005651' 260 17 0 00 262202

005652' 262 17 0 00 000004              GET     AC4                     ;Restore the ac's
005653' 262 17 0 00 000003              GET     AC3
005654' 262 17 0 00 000002              GET     AC2
005655' 262 17 0 00 000001              GET     AC1
005656' 263 17 0 00 000000              RTN                             ;Thats it

                                ;Return local node number(.Ssgln) argument block

005657' 000000  000002          LOCARG: 0,,.LBGLN                       ;Processed words,,length of block
005660' 000 00 0 00 000000              Z                               ;Local ci node number
```

```
                                        SUBTTL  Return Buffer Sizes

                                ;This routine returns the largest "DATAGRAM" & "MESSAGE"
                                ;That can be sent on this node.
                                ;First we set up the argument block then execute the scs jsys.
                                ;Call:
                                ;       Go      buffsz                  ;call this routine
                                ;       +1                              ;Return here always
                                ;
005661' 261 17 0 00 000001      BUFFSZ: PUT     AC1                     ;Save the ac's
005662' 261 17 0 00 000002              PUT     AC2
005663' 261 17 0 00 000003              PUT     AC3
005664' 261 17 0 00 000004              PUT     AC4
005665' 201 04 0 00 000003              MOVEI   AC4,.LBRBS              ;Get the length of arg block
005666' 202 04 0 00 005704'             MOVEM   AC4,.SQLEN+BUFARG       ;Store length of argument block
005667' 402 00 0 00 005705'             SETZM   .SQLMG+BUFARG           ;Clear the length of "MSG" word
005670' 402 00 0 00 005706'             SETZM   .SQLDG+BUFARG           ;Clear the length of "DGM" word
005671' 201 01 0 00 000035              MOVEI   AC1,.SSRBS              ;Return max buffer sizes function code
005672' 201 02 0 00 005704'             MOVEI   AC2,BUFARG              ;Address of argument block to ac2
005673' 260 17 0 00 004605'             GO      DOSCS                   ;Execute the jsys
005674' 260 17 0 00 004633'              GO     ERRSCS                  ;If error go here

005675' 332 00 0 00 000216'             TRMSG   (Return Buffer Sizes)
005676' 260 17 0 00 262214

005677' 262 17 0 00 000004              GET     AC4                     ;Restore the ac's
005700' 262 17 0 00 000003              GET     AC3
005701' 262 17 0 00 000002              GET     AC2
005702' 262 17 0 00 000001              GET     AC1
005703' 263 17 0 00 000000              RTN                             ;Thats it

                                ;Return maximum datagram & message transfer sizes (.Ssrbs) argument block

005704' 000000  000003          BUFARG: 0,,.LBRBS                       ;Processed words,,length of block
005705' 000 00 0 00 000000              Z                               ;Size (words) of max "MSG" buffer
005706' 000 00 0 00 000000              Z                               ;Size (words) of max "DGM" buffer
```

```
                                        SUBTTL  Return Status

                                ;This routine returns some information about a connection.
                                ;First we set up the argument block then execute the scs jsys.
                                ;Call:
                                ;       Move    ac1,[n]                 ;n=connect id
                                ;       Go      statsi                  ;call this routine
                                ;       +1                              ;Return here always

005707' 261 17 0 00 000001      STATSI: PUT     AC1                     ;Save the ac's
005710' 261 17 0 00 000002              PUT     AC2
005711' 261 17 0 00 000003              PUT     AC3
005712' 261 17 0 00 000004              PUT     AC4
005713' 201 04 0 00 000004              MOVEI   AC4,.LBSTS              ;Load length
005714' 202 04 0 00 005733'             MOVEM   AC4,.SQLEN+STAARG       ;Store length of argument block
005715' 202 01 0 00 005734'             MOVEM   AC1,.SQCID+STAARG       ;Store the connect id
005716' 402 00 0 00 005735'             SETZM   .SQFST+STAARG           ;Clear the flag word
005717' 402 00 0 00 005736'             SETZM   .SQSBR+STAARG           ;Clear the node# word
005720' 201 01 0 00 000012              MOVEI   AC1,.SSSTS              ;Return status info function code
005721' 201 02 0 00 005733'             MOVEI   AC2,STAARG              ;Address of argument block to ac2
005722' 260 17 0 00 004605'             GO      DOSCS                   ;Execute the jsys
005723' 260 17 0 00 004633'              GO     ERRSCS                  ;If error go here

005724' 332 00 0 00 000216'             TRMSG   (Return Status)
005725' 260 17 0 00 262225

005726' 262 17 0 00 000004              GET     AC4                     ;Restore the ac's
005727' 262 17 0 00 000003              GET     AC3
005730' 262 17 0 00 000002              GET     AC2
005731' 262 17 0 00 000001              GET     AC1
005732' 263 17 0 00 000000              RTN                             ;Thats it

                                ;Return status information(.Sssts) argument block

005733' 000000  000004          STAARG: 0,,.LBSTS                       ;Processed words,,length of block
005734' 000 00 0 00 000000              Z                               ;Connect id
005735' 000 00 0 00 000000              Z                               ;Flags,,connect state
005736' 000 00 0 00 000000              Z                               ;Node# of remote
```

```
                                        SUBTTL  Return Configuration Information

                                ;This routine returns information about another system.
                                ;First we set up the argument block then execute the scs jsys.
                                ;Call:
                                ;       Move    ac1,[n]                 ;n=connect id
                                ;       Move    ac2,[m]                 ;m=remote node#
                                ;       Go      config                  ;call this routine
                                ;       +1                              ;Error return
                                ;       +2                              ;Good return
                                ;

005737' 261 17 0 00 000001      CONFIG: PUT     AC1                     ;Save the ac's
005740' 261 17 0 00 000002              PUT     AC2
005741' 261 17 0 00 000003              PUT     AC3
005742' 261 17 0 00 000004              PUT     AC4
005743' 201 04 0 00 000024              MOVEI   AC4,.LBRCD              ;Load length
005744' 202 04 0 00 005767'             MOVEM   AC4,.SQLEN+CNFARG       ;Store length of argument block
005745' 202 01 0 00 005770'             MOVEM   AC1,.SQCID+CNFARG       ;Store the connect id
005746' 202 02 0 00 005771'             MOVEM   AC2,.SQOSB+CNFARG       ;Store the remote node#
005747' 402 00 0 00 005772'             SETZM   .SQVCS+CNFARG           ;Clear previous
005750' 505 01 0 00 005772'             HRLI    AC1,.SQVCS+CNFARG       ; Information.
005751' 541 01 0 00 005773'             HRRI    AC1,.SQSAD+CNFARG
005752' 251 01 0 00 006012'             BLT     AC1,CNFARG+<.LBRCD-1>
005753' 201 01 0 00 000011              MOVEI   AC1,.SSRCD              ;Configuration function code
005754' 201 02 0 00 005767'             MOVEI   AC2,CNFARG              ;Address of argument block to ac2
005755' 260 17 0 00 004605'             GO      DOSCS                   ;Execute the jsys
005756' 254 00 0 00 005760'              JRST   CONFIX                  ;If error go here(+1 rtn)
005757' 350 00 0 17 777774              AOS     -4(P)                   ;+2 Good return

005760' 332 00 0 00 000216'     CONFIX: TRMSG   (Configuration)
005761' 260 17 0 00 262236
005762' 262 17 0 00 000004              GET     AC4                     ;Restore the ac's
005763' 262 17 0 00 000003              GET     AC3
005764' 262 17 0 00 000002              GET     AC2
005765' 262 17 0 00 000001              GET     AC1
005766' 263 17 0 00 000000              RTN                             ;Thats it
```

```
4570
4571                                    ;Configuration data(.Ssrcd) argument block
4572
4573  005767' 000000  000024           CNFARG: 0,,.LBRCD                       ;Processed words,,length of block
4574  005770' 000 00 0 00 000000              Z                                 ;Optional connect id
4575  005771' 000 00 0 00 000000              Z                                 ;Optional remote node#
4576  005772' 000 00 0 00 000000              Z                                 ;Virtual circuit state,, node number
4577  005773' 000 00 0 00 000000              Z                                 ;System address
4578  005774' 000 00 0 00 000000              Z                                 ; (6 8-Bit bytes)
4579  005775' 000 00 0 00 000000              Z                                 ;Max dest datagram size
4580  005776' 000 00 0 00 000000              Z                                 ;Max dest message size
4581  005777' 000 00 0 00 000000              Z                                 ;Dest software type
4582  006000' 000 00 0 00 000000              Z                                 ;Dest software version
4583  006001' 000 00 0 00 000000              Z                                 ;Dest software edit level
4584  006002' 000 00 0 00 000000              Z                                 ;Dest hardware type (2 words)
4585  006003' 000 00 0 00 000000              Z
4586
4587  006004' 000 00 0 00 000000              Z                                 ;Dest hardware version
4588
4589  006005' 000 00 0 00 000000              Z                                 ;Dest port name (3 words)
4590  006006' 000 00 0 00 000000              Z                                 ;
4591  006007' 000 00 0 00 000000              Z                                 ;
4592
4593  006010' 000 00 0 00 000000              Z                                 ;Port characteristics (2 words)
4594  006011' 000 00 0 00 000000              Z                                 ; (8-Bit bytes)
4595
4596  006012' 000 00 0 00 000000              Z                                 ;Local port number (ci#)
```

```
                                        SUBTTL  Map Buffer

                                ;This routine gets a buffer name from sca to be used in dma transfers.
                                ;First we set up the argument block then execute the scs jsys.
                                ;Call:
                                ;       Move    ac1,[n]                 ;n=memory address(e.G. Buf1)
                                ;       Move    ac2,[m]                 ;m=length of memory block
                                ;       Go      getbuf                  ;call this routine
                                ;       +1                              ;Error return
                                ;       +2                              ;Ac1 = returned 32 bit buffer name

006013' 261 17 0 00 000000      GETBUF: PUT     AC0                     ;Save ac
006014' 201 00 0 00 000005              MOVEI   AC0,5                   ;Default block length
006015' 202 00 0 00 006033'             MOVEM   AC0,.SQLEN+GETARG       ;Store length of argument block
006016' 202 02 0 00 006036'             MOVEM   AC2,.SQBNA+1+GETARG     ;Store the length of memory block
006017' 202 01 0 00 006037'             MOVEM   AC1,.SQBNA+2+GETARG     ;Store the memory address
006020' 402 00 0 00 006035'             SETZM   .SQBNA+GETARG           ;Clear the last buffer name
006021' 201 01 0 00 000014              MOVEI   AC1,.SSMAP              ;Map a buffer function code
006022' 201 02 0 00 006033'             MOVEI   AC2,GETARG              ;Address of argument block to ac2
006023' 260 17 0 00 004605'             GO      DOSCS                   ;Execute the jsys
006024' 254 00 0 00 006027'              JRST   GETBUA                  ;If error do +1 return
006025' 200 01 0 00 006035'             MOVE    AC1,.SQBNA+GETARG       ;Get the buffer name in ac1
006026' 350 00 0 17 777777              AOS     -1(P)                   ;+2 Good return
006027' 262 17 0 00 000000      GETBUA: GET     AC0                     ;Restore ac

006030' 332 00 0 00 000216'             TRMSG   (Map Buffer)
006031' 260 17 0 00 262246
006032' 263 17 0 00 000000              RTN                             ;Thats it

                                ;Map a buffer (.Ssmap) argument block

006033' 000000  000005          GETARG: 0,,5                            ;Processed words,,length of block
006034' 000000  000010                  SQ%CVD                          ;Flags-Do not clr valid bit.
006035' 000 00 0 00 000000              Z                               ;Returned 32 bit buffer name
006036' 000 00 0 00 000000              Z                               ;Length of memory block
006037' 000 00 0 00 000000              Z                               ;Memory address(e.G. Buf1)
```

```
                                        SUBTTL  Unmap Buffer

                                ;This routine removes(unmaps) a buffer from sca, used in dma transfers.
                                ;First we set up the argument block then execute the scs jsys.
                                ;Call:
                                ;       Move    ac1,[n]                 ;n=32 bit buffer name
                                ;       Go      rmvbuf                  ;call this routine
                                ;       +1                              ;Good return
                                ;       +2                              ;Error return

006040' 261 17 0 00 000001      RMVBUF: PUT     AC1                     ;Save the ac's
006041' 261 17 0 00 000002              PUT     AC2
006042' 261 17 0 00 000003              PUT     AC3
006043' 261 17 0 00 000004              PUT     AC4
006044' 201 02 0 00 000002              MOVEI   AC2,.LBUMP              ;Load length
006045' 202 02 0 00 006064'             MOVEM   AC2,.SQLEN+RMVARG       ;Store length of argument block
006046' 202 01 0 00 006065'             MOVEM   AC1,.SQNAM+RMVARG       ;Store the memory address
006047' 201 01 0 00 000015              MOVEI   AC1,.SSUMP              ;Unmap a buffer function code
006050' 201 02 0 00 006064'             MOVEI   AC2,RMVARG              ;Address of argument block to ac2
006051' 260 17 0 00 004605'             GO      DOSCS                   ;Execute the jsys
006052' 254 00 0 00 006054'              JRST   RMVBUA                  ;Error go here
006053' 334 00 0 00 000000              SKIPA                           ;Take +1 good return
006054' 350 00 0 17 777774      RMVBUA: AOS     -4(P)                   ;Take +2 error return
006055' 262 17 0 00 000004              GET     AC4                     ;Restore the ac's
006056' 262 17 0 00 000003              GET     AC3
006057' 262 17 0 00 000002              GET     AC2
006060' 262 17 0 00 000001              GET     AC1

006061' 332 00 0 00 000216'             TRMSG   (Unmap Buffer)
006062' 260 17 0 00 262256

006063' 263 17 0 00 000000              RTN                             ;Thats it

                                ;Unmap a buffer(.Ssump) argument block

006064' 000000  000002          RMVARG: 0,,.LBUMP                       ;Processed words,,length of block
006065' 000 00 0 00 000000              Z                               ;32 Bit buffer name to be unmapped
```

```
                                        SUBTTL  Send Data

                                ;This routine transfers data to a remote node.
                                ;First we set up the argument block then execute the scs jsys.
                                ;Call:
                                ;       Move    ac1,[n]                 ;n=cid
                                ;       Move    ac2,[m]                 ;m=32 bit send buffer name
                                ;       Move    ac3,[y]                 ;y=32 bit receive buffer name
                                ;       Move    ac4,[z]                 ;z=xmit offset,,receive offset
                                ;       Go      sedata                  ;call this routine
                                ;       +1                              ;Return here always

006066' 261 17 0 00 000005      SEDATA: PUT     AC5                     ;Save the ac's
006067' 261 17 0 00 000006              PUT     AC6
006070' 201 06 0 00 000005              MOVEI   AC6,.LBSND              ;Load length
006071' 202 06 0 00 006110'             MOVEM   AC6,.SQLEN+SEDARG       ;Store length of argument block
006072' 202 01 0 00 006111'             MOVEM   AC1,.SQCID+SEDARG       ;Store the cid
006073' 202 02 0 00 006112'             MOVEM   AC2,.SQSNM+SEDARG       ;Store the send buffer name
006074' 202 03 0 00 006113'             MOVEM   AC3,.SQRNM+SEDARG       ;Store the receive buffer name
006075' 202 04 0 00 006114'             MOVEM   AC4,.SQOFS+SEDARG       ;Store the offsets
006076' 201 01 0 00 000016              MOVEI   AC1,.SSSND              ;Send data function code
006077' 201 02 0 00 006110'             MOVEI   AC2,SEDARG              ;Address of argument block to ac2
006100' 260 17 0 00 004605'             GO      DOSCS                   ;Execute the jsys
006101' 260 17 0 00 004633'              GO     ERRSCS                  ;If error go here
006102' 350 00 0 00 000261'             AOS     DMAWRT                  ;Count of dma writes

006103' 332 00 0 00 000216'             TRMSG   (Send Data)
006104' 260 17 0 00 262266

006105' 262 17 0 00 000006              GET     AC6                     ;Restore the ac's
006106' 262 17 0 00 000005              GET     AC5
006107' 263 17 0 00 000000              RTN                             ;Thats it

                                ;Send data (.Sssnd) argument block

006110' 000000  000005          SEDARG: 0,,.LBSND                       ;Processed words,,length of block
006111' 000 00 0 00 000000              Z                               ;Cid
006112' 000 00 0 00 000000              Z                               ;32 Bit buffer name of send buffer
006113' 000 00 0 00 000000              Z                               ;32 Bit buffer name of receive buffer
006114' 000 00 0 00 000000              Z                               ;Xmit offset,,receive offset
```

```
                                        SUBTTL  Read Data

                                ;This routine tells sca that data is to be expected for the given buffer name.
                                ;First we set up the argument block then execute the scs jsys.
                                ;Call:
                                ;       Move    ac1,[n]                 ;n=cid
                                ;       Move    ac2,[m]                 ;m=32 bit send buffer name
                                ;       Move    ac3,[y]                 ;y=32 bit receive buffer name
                                ;       Move    ac4,[z]                 ;z=xmit offset,,receive offset
                                ;       Go      redata                  ;call this routine
                                ;       +1                              ;Return here always

006115' 261 17 0 00 000005      REDATA: PUT     AC5                     ;Save the ac's
006116' 261 17 0 00 000006              PUT     AC6
006117' 201 06 0 00 000005              MOVEI   AC6,.LBREQ              ;Load length
006120' 202 06 0 00 006137'             MOVEM   AC6,.SQLEN+REDARG       ;Store length of argument block
006121' 202 01 0 00 006140'             MOVEM   AC1,.SQCID+REDARG       ;Store the cid
006122' 202 02 0 00 006141'             MOVEM   AC2,.SQSNM+REDARG       ;Store the send buffer name
006123' 202 03 0 00 006142'             MOVEM   AC3,.SQRNM+REDARG       ;Store the receive buffer name
006124' 202 04 0 00 006143'             MOVEM   AC4,.SQOFS+REDARG       ;Store the offsets
006125' 201 01 0 00 000017              MOVEI   AC1,.SSREQ              ;Request data function code
006126' 201 02 0 00 006137'             MOVEI   AC2,REDARG              ;Address of argument block to ac2
006127' 260 17 0 00 004605'             GO      DOSCS                   ;Execute the jsys
006130' 260 17 0 00 004633'              GO     ERRSCS                  ;If error go here
006131' 350 00 0 00 000262'             AOS     DMARD                   ;Count of dma reads

006132' 332 00 0 00 000216'             TRMSG   (Read Data)
006133' 260 17 0 00 262276

006134' 262 17 0 00 000006              GET     AC6                     ;Restore the ac's
006135' 262 17 0 00 000005              GET     AC5
006136' 263 17 0 00 000000              RTN                             ;Thats it

                                ;Request data(.Ssreq) argument block

006137' 000000  000005          REDARG: 0,,.LBREQ                       ;Processed words,,length of block
006140' 000 00 0 00 000000              Z                               ;Cid
006141' 000 00 0 00 000000              Z                               ;32 Bit buffer name of send buffer
006142' 000 00 0 00 000000              Z                               ;32 Bit buffer name of receive buffer
006143' 000 00 0 00 000000              Z                               ;Xmit offset,,receive offset
```

```
                                        SUBTTL  Get Data Queue Entry

                                ;Get entry from the data queue(dma).
                                ;We come here only during interrupt time.
                                ;Call:
                                ;       Go      dataq                   ;call this routine
                                ;       +1                              ;Ac3=buffer name
                                ;                                       ;If no more events ac3 = 0

006144' 261 17 0 00 000001      DATAQ:  PUT     AC1                     ;Save the ac's
006145' 261 17 0 00 000002              PUT     AC2
006146' 261 17 0 00 000004              PUT     AC4
006147' 201 01 0 00 000003              MOVEI   AC1,.LBGDE              ;Set up length of
006150' 202 01 0 00 006170'             MOVEM   AC1,.SQLEN+DATARG       ; The argument block
006151' 476 00 0 00 006171'             SETOM   .SQCID+DATARG           ;Use any CID
006152' 402 00 0 00 006172'             SETZM   .SQBID+DATARG           ;Clear the buffer id location
006153' 201 01 0 00 000024              MOVEI   AC1,.SSGDE              ;Function code
006154' 201 02 0 00 006170'             MOVEI   AC2,DATARG              ;Address of argument block
006155' 260 17 0 00 004605'             GO      DOSCS                   ;Execute the jsys
006156' 254 00 0 00 006161'              JRST   DATATE                  ;If error go here(no more events)
006157' 200 03 0 00 006172'             MOVE    AC3,.SQBID+DATARG       ;Get the buffer id
006160' 334 00 0 00 000000              SKIPA
006161' 402 00 0 00 000003      DATATE: SETZM   AC3                     ;Flag the error
006162' 262 17 0 00 000004              GET     AC4                     ;Restore the ac's
006163' 262 17 0 00 000002              GET     AC2
006164' 262 17 0 00 000001              GET     AC1

006165' 332 00 0 00 000216'             TRMSG   (Get Data Queue Entry)
006166' 260 17 0 00 262310
006167' 263 17 0 00 000000              RTN                             ;Thats it

006170' 000000  000003          DATARG: 0,,.LBGDE                       ;Words processed,, length of block
006171' 000 00 0 00 000000              Z                               ;Connect id
006172' 000 00 0 00 000000              Z                               ;Buffer id whos xfer completed
```

```
                                        SUBTTL  Get Event Queue Entry

                                ;Get entry from the event queue.
                                ;We come here only during interrupt time.
                                ;Call:
                                ;       Go      eventq                  ;call this routine
                                ;       +1                              ;Ac3 = cid  &  ac4 = event code
                                ;                                       ;Ac4 = 0 if no more events
                                ;

006173' 261 17 0 00 000001      EVENTQ: PUT     AC1                     ;Save the ac's
006174' 261 17 0 00 000002              PUT     AC2
006175' 261 17 0 00 000005              PUT     AC5
006176' 201 01 0 00 000010              MOVEI   AC1,.LBEVT              ;Set up length of
006177' 202 01 0 00 006237'             MOVEM   AC1,.SQLEN+EVTARG       ; The argument block
006200' 476 00 0 00 006240'             SETOM   .SQCID+EVTARG           ;-1 = Get any msg for this fork
006201' 402 00 0 00 006241'             SETZM   .SQESB+EVTARG           ;Clear the 1st location for the blt.
006202' 505 01 0 00 006241'             HRLI    AC1,.SQESB+EVTARG       ;Zero out
006203' 541 01 0 00 006242'             HRRI    AC1,.SQEVT+EVTARG       ; The argument block
006204' 251 01 0 00 006246'             BLT     AC1,EVTARG+<.LBEVT-1>   ; For incoming data
006205' 201 01 0 00 000025              MOVEI   AC1,.SSEVT              ;Function code
006206' 201 02 0 00 006237'             MOVEI   AC2,EVTARG              ;Address of argument block
006207' 260 17 0 00 004605'             GO      DOSCS                   ;Execute the jsys
006210' 254 00 0 00 006214'              JRST   EVENTE                  ;If error go here(no more events)
006211' 200 03 0 00 006240'             MOVE    AC3,.SQCID+EVTARG       ;Get the cid
006212' 200 04 0 00 006242'             MOVE    AC4,.SQEVT+EVTARG       ;Get the event code
006213' 334 00 0 00 000000              SKIPA
006214' 402 00 0 00 000004      EVENTE: SETZM   AC4                     ;Flag the error
006215' 262 17 0 00 000005              GET     AC5                     ;Restore the ac's
006216' 262 17 0 00 000002              GET     AC2
006217' 262 17 0 00 000001              GET     AC1
```

```
4815
4816                                    ;Output trace message if needed
4817
4818  006220' 332 00 0 00 000004'               SKIPE   AC4                     ;If no more events don't type
4819  006221' 336 00 0 00 000216'               SKIPN   TRACE                   ;Are we in trace mode?
4820  006222' 254 00 0 00 006236'               JRST    EVENTX                  ;No, don't type anything
4821  006223' 261 17 0 00 000001                PUT     AC1
4822  006224' 261 17 0 00 000002                PUT     AC2
4823  006225' 261 17 0 00 000003                PUT     AC3
4824  006226' 561 01 0 00 262315                HRROI   AC1,[ASCIZ/Get event /]
4825  006227' 260 17 0 00 004550'               GO      PRINTT                  ;Print that
4826  006230' 200 02 0 00 000004                MOVE    AC2,AC4                 ;Get the event code
4827  006231' 260 17 0 00 004564'               GO      PRINTD                  ;Print it in decimal
4828  006232' 260 17 0 00 004547'               GO      PRINTC                  ;Print CRLF
4829  006233' 262 17 0 00 000003                GET     AC3
4830  006234' 262 17 0 00 000002                GET     AC2
4831  006235' 262 17 0 00 000001                GET     AC1
4832
4833  006236' 263 17 0 00 000000        EVENTX: RTN                             ;Thats it
4834
4835  006237' 000000  000010            EVTARG: 0,,.LBEVT                       ;Words processed,, length of block
4836  006240' 000 00 0 00 000000                Z                               ;Cid
4837  006241' 000 00 0 00 000000                Z                               ;Node# of remote node
4838  006242' 000 00 0 00 000000                Z                               ;Event code
4839  006243' 000 00 0 00 000000                Z                               ;Event data
4840  006244' 000 00 0 00 000000                Z                               ;Event data
4841  006245' 000 00 0 00 000000                Z                               ;Event data
4842  006246' 000 00 0 00 000000                Z                               ;Event data
```

```
                                        SUBTTL  Perform DIAG JSYS or UUO

                                ;This routine executes the "DIAG" jsys which is used to send either a "START"
                                ;Or "STOP" packet on the ci or to "START" or "STOP" the poller.
                                ;Call:
                                ;       Set up the "DIAGAG" argument block before calling this routine.
                                ;       Move    ac1,[n]                 ;n = -length,,address of argument block
                                ;       Go      diagj                   ;call this routine
                                ;       +1                              ;Error return
                                ;       +2                              ;Good return

006247' 261 17 0 00 000001      DIAGJ:  PUT     AC1                     ;Save acs
006250' 261 17 0 00 000002              PUT     AC2
006251' 261 17 0 00 000003              PUT     AC3
006252' 261 17 0 00 000004              PUT     AC4
006253' 201 04 0 01 000000              MOVEI   AC4,(AC1)               ;Load the address of the argument blk
006254' 336 00 0 00 000234'             SKIPN   MONTYP                  ;Skip if TOPS-20
006255' 254 00 0 00 006261'             JRST    DIAGU                   ;TOPS-10, DIAG UUO
006256' 104 00 0 00 000530              DIAG%                           ;The DIAG JSYS
006257' 320 16 0 00 006271'              ERJMP  ERRDI2                  ;Error call(for trace only)
006260' 254 00 0 00 006263'             JRST    DIAGS                   ;Success

006261' 047 01 0 00 000163      DIAGU:  DIAG.   AC1,                    ;The UUO
006262' 254 00 0 00 006275'              JRST   ERRDI1                  ;Report error

006263' 350 00 0 17 777774      DIAGS:  AOS     -4(P)                   ;Good return
006264' 262 17 0 00 000004      DIAGF:  GET     AC4                     ;Restore the acs
006265' 262 17 0 00 000003              GET     AC3
006266' 262 17 0 00 000002              GET     AC2
006267' 262 17 0 00 000001              GET     AC1
006270' 263 17 0 00 000000              RTN                             ;Return
```

```
4874
4875                                    ;Here of error on the -20
4876
4877  006271' 561 01 0 00 262320  ERRDI2: HRROI   AC1,[ASCIZ/? DIAG JSYS error: /] ;Load error text
4878  006272' 260 17 0 00 004550'         GO      PRINTT                          ;Print it
4879  006273' 260 17 0 00 004676'         GO      ERSTRT                          ;Print error string to terminal
4880  006274' 254 00 0 00 006313'         JRST    ERRDI4                          ;Print PC, etc
4881
4882                                    ;Here if error on the -10
4883
4884  006275' 261 17 0 00 000001  ERRDI1: PUT     AC1                             ;Save error code
4885  006276' 561 01 0 00 262324          HRROI   AC1,[ASCIZ/? DIAG UUO error: (/] ;Load error text
4886  006277' 260 17 0 00 004550'         GO      PRINTT                          ;Output that
4887  006300' 200 02 0 17 000000          MOVE    AC2,(P)                         ;Load error code
4888  006301' 260 17 0 00 004555'         GO      PRINTO                          ;Print in octal
4889  006302' 561 01 0 00 261464          HRROI   AC1,[ASCIZ/) /]                 ;Load finish text
4890  006303' 260 17 0 00 004550'         GO      PRINTT                          ;Print that
4891  006304' 262 17 0 00 000002          GET     AC2                             ;Get error code back in AC2
4892
4893  006305' 205 01 0 00 777765          MOVSI   AC1,-ERRDIL                     ;Load length of table
4894  006306' 554 03 0 01 006326' ERRDI3: HLRZ    AC3,ERRDIT(AC1)                 ;Load an error code
4895  006307' 302 02 0 03 000000          CAIE    AC2,(AC3)                       ;Skip if match
4896  006310' 253 01 0 00 006306'         AOBJN   AC1,ERRDI3                      ;Loop for more looking
4897  006311' 560 01 0 01 006326'         HRRO    AC1,ERRDIT(AC1)                 ;Point to asciz text
4898  006312' 260 17 0 00 004550'         GO      PRINTT                          ;Output that please
4899
4900  006313'                     ERRDI4: HRROI   AC1,[ASCIZ/
4901  006313' 561 01 0 00 262330  Function /]                                     ;Load string to print
4902  006314' 550 02 0 04 000000          HRRZ    AC2,(AC4)                       ;Load the function code
4903  006315' 260 17 0 00 004550'         GO      PRINTT                          ;Print that
4904  006316' 260 17 0 00 004555'         GO      PRINTO                          ;Print that in octal please
4905  006317' 561 01 0 00 261475          HRROI   AC1,[ASCIZ/, called from PC /] ;Print a label for the next bit
4906  006320' 260 17 0 00 004550'         GO      PRINTT                          ;Print that
4907  006321' 550 02 0 17 777774          HRRZ    AC2,-4(P)                       ;Get pc+1
4908  006322' 275 02 0 00 000001          SUBI    AC2,1                           ;Point to right location
4909  006323' 260 17 0 00 004555'         GO      PRINTO                          ;Print octal number
4910  006324' 260 17 0 00 004547'         GO      PRINTC                          ;Print CRLF
4911  006325' 254 00 0 00 006264'         JRST    DIAGF                           ;Failed, return
```

```
                                        ;TOPS-10 error table

006326' 000001  262333          ERRDIT: XWD     DIANP%,[ASCIZ/Not enough privileges/]
006327' 000002  262340                  XWD     DIAIA%,[ASCIZ/Illegal number of arguments/]
                                        ;XWD    DIAIC%,[ASCIZ/Illegal controller number/]
                                        ;XWD    DIAIU%,[ASCIZ/Illegal unit number/]
                                        ;XWD    DIAAA%,[ASCIZ/Some units already assigned/]
                                        ;XWD    DIADM%,[ASCIZ/Unit not in diagnostic mode/]
006330' 000007  262346                  XWD     DIAAJ%,[ASCIZ/Unit assigned to another job/]
006331' 000010  262354                  XWD     DIAFC%,[ASCIZ/Not enough free core/]
                                        ;XWD    DIAAU%,[ASCIZ/No assigned units/]
                                        ;XWD    DIACP%,[ASCIZ/IOWD crosses page boundry/]
006332' 000013  262361                  XWD     DIAIF%,[ASCIZ/Illegal function/]
006333' 000014  262365                  XWD     DIAVC%,[ASCIZ/Job must not be virtual/]
006334' 000015  262372                  XWD     DIANC%,[ASCIZ/No such CPU/]
006335' 000016  262375                  XWD     DIANR%,[ASCIZ/CPU not running/]
006336' 000017  262401                  XWD     DIABA%,[ASCIZ/Bad argument list/]
006337' 000020  261550                  XWD     DIACI%,[ASCIZ/No CI port on specified CPU/]
006340' 000021  262405                  XWD     DIATO%,[ASCIZ/Read port counters function timed out/]
                                        ;XWD    DIANK%,[ASCIZ/No NI port on specified CPU/]
                                        ;XWD    DIARF%,[ASCIZ/Microcode reload failed/]
006341' 777777  262415                  XWD     777777,[ASCIZ/Unknown DIAG error code/]
                000013                  ERRDIL==.-ERRDIT-1

006342' 000 00 0 00 000000      DIAGAG: Z                       ;Diag function code
006343' 000 00 0 00 000000              Z                       ;Argument channel,,counter function
006344' 000 00 0 00 000000              Z                       ;Argument counter,,fork number
006345' 000 00 0 00 000000              Z                       ;Microcode version
006346' 000 00 0 00 000000              Z                       ;Path a ack count
006347' 000 00 0 00 000000              Z                       ;Path a nak count
006350' 000 00 0 00 000000              Z                       ;Path a no response count
006351' 000 00 0 00 000000              Z                       ;Path b ack count
006352' 000 00 0 00 000000              Z                       ;Path b nak count
006353' 000 00 0 00 000000              Z                       ;Path b no response count
006354' 000 00 0 00 000000              Z                       ;Number of dropped datagrams
006355' 000 00 0 00 000000              Z                       ;# Of packets transmitted
006356' 000 00 0 00 000000              Z                       ;# Of packets received
006357' 000 00 0 00 000000              Z                       ;Designated port for counters
006360' 000 00 0 00 000000              Z                       ;Monitor stuff
006361' 000 00 0 00 000000              Z                       ;Monitor stuff
006362' 000 00 0 00 000000              Z                       ;Monitor stuff
006363' 000 00 0 00 000000              Z                       ;Monitor stuff
006364' 000 00 0 00 000000              Z                       ;Monitor stuff
006365' 000 00 0 00 000000              Z                       ;Monitor stuff
006366'                         Block 4                         ;Some extra nothings
```

```
                                        SUBTTL  TOPS-10 Interrupt Dispatch

                                ;Here on a SCS event interrupt for TOPS-10

006372' 261 17 0 00 000000      INTTEN: PUT     AC0                     ;Save the ACs
006373' 261 17 0 00 000001              PUT     AC1
006374' 261 17 0 00 000002              PUT     AC2
006375' 261 17 0 00 000003              PUT     AC3
006376' 261 17 0 00 000004              PUT     AC4
006377' 261 17 0 00 000005              PUT     AC5
006400' 261 17 0 00 000006              PUT     AC6
                                        MOVEI   AC2,[   TRMSG (Unknown interrupt cause)
006401' 201 02 0 00 262435                              RTN]            ;Load default routine
006402' 200 01 0 00 007312'             MOVE    AC1,PIVECT+SCOFFS+.PSVIS ;Get interrupt status word
006403' 603 01 0 00 400000              TLNE    AC1,(SC%MSA)            ;Message available?
006404' 201 02 0 00 006574'             MOVEI   AC2,INPMSG              ;Yes
006405' 603 01 0 00 200000              TLNE    AC1,(SC%DGA)            ;Datagram available?
006406' 201 02 0 00 006477'             MOVEI   AC2,INPDGM              ;Yep
006407' 603 01 0 00 100000              TLNE    AC1,(SC%DTA)            ;DMA transfer complete
006410' 201 02 0 00 006720'             MOVEI   AC2,DMAXFR              ;Yep
006411' 603 01 0 00 040000              TLNE    AC1,(SC%EVA)            ;Other event pending?
006412' 201 02 0 00 006761'             MOVEI   AC2,OTHREV              ;Yep
006413' 260 17 0 02 000000              GO      (AC2)                   ;Dispatch to proper routine
006414' 262 17 0 00 000006              GET     AC6                     ;Restore the acs
006415' 262 17 0 00 000005              GET     AC5
006416' 262 17 0 00 000004              GET     AC4
006417' 262 17 0 00 000003              GET     AC3
006420' 262 17 0 00 000002              GET     AC2
006421' 262 17 0 00 000001              GET     AC1
006422' 262 17 0 00 000000              GET     AC0
006423' 047 00 0 00 000137              DEBRK.                          ;Dismiss the interrupt
```

```
                                        SUBTTL  TOPS-20 Interrupt Dispatch

                                ;Here for TOPS-20 interrupt system dispatch

006424' 261 17 0 00 000000      INPDGO: PUT     AC0                     ;Save the ACs
006425' 261 17 0 00 000001              PUT     AC1
006426' 261 17 0 00 000002              PUT     AC2
006427' 261 17 0 00 000003              PUT     AC3
006430' 261 17 0 00 000004              PUT     AC4
006431' 261 17 0 00 000005              PUT     AC5
006432' 261 17 0 00 000006              PUT     AC6
006433' 260 17 0 00 006477'             GO      INPDGM                  ;Channel 23 datagram available
006434' 254 00 0 00 006467'             JRST    INTDMS                  ;Dismiss interrupt

006435' 261 17 0 00 000000      INPMSO: PUT     AC0                     ;Save the ACs
006436' 261 17 0 00 000001              PUT     AC1
006437' 261 17 0 00 000002              PUT     AC2
006440' 261 17 0 00 000003              PUT     AC3
006441' 261 17 0 00 000004              PUT     AC4
006442' 261 17 0 00 000005              PUT     AC5
006443' 261 17 0 00 000006              PUT     AC6
006444' 260 17 0 00 006574'             GO      INPMSG                  ;Channel 24 message available
006445' 254 00 0 00 006467'             JRST    INTDMS                  ;Dismiss interrupt

006446' 261 17 0 00 000000      DMAXFO: PUT     AC0                     ;Save the ACs
006447' 261 17 0 00 000001              PUT     AC1
006450' 261 17 0 00 000002              PUT     AC2
006451' 261 17 0 00 000003              PUT     AC3
006452' 261 17 0 00 000004              PUT     AC4
006453' 261 17 0 00 000005              PUT     AC5
006454' 261 17 0 00 000006              PUT     AC6
006455' 260 17 0 00 006720'             GO      DMAXFR                  ;Channel 25 - dma xfer complete
006456' 254 00 0 00 006467'             JRST    INTDMS                  ;Dismiss interrupt
```

```
5022
5023  006457' 261 17 0 00 000000      OTHREO: PUT     AC0                     ;Save the ACs
5024  006460' 261 17 0 00 000001              PUT     AC1
5025  006461' 261 17 0 00 000002              PUT     AC2
5026  006462' 261 17 0 00 000003              PUT     AC3
5027  006463' 261 17 0 00 000004              PUT     AC4
5028  006464' 261 17 0 00 000005              PUT     AC5
5029  006465' 261 17 0 00 000006              PUT     AC6
5030  006466' 260 17 0 00 006761'              GO      OTHREV                  ;Channel 26 for connections & errors
5031
5032                                    ;Here to dismiss interupts
5033
5034  006467' 262 17 0 00 000006      INTDMS: GET     AC6                     ;Restore the acs
5035  006470' 262 17 0 00 000005              GET     AC5
5036  006471' 262 17 0 00 000004              GET     AC4
5037  006472' 262 17 0 00 000003              GET     AC3
5038  006473' 262 17 0 00 000002              GET     AC2
5039  006474' 262 17 0 00 000001              GET     AC1
5040  006475' 262 17 0 00 000000              GET     AC0
5041  006476' 104 00 0 00 000136              DEBRK%                          ;Dismiss interrupt
```

```
                                        SUBTTL  Interrupt Routine For Inputting Datagrams


                                ;Here on a message available datagram

006477' 332 00 0 00 000216'     INPDGM: TRMSG   (Interrupt for Input Datagram)

006500' 260 17 0 00 262447
006501' 260 17 0 00 005452'     INPDGH: GO      RCVDGM                  ;Receive a datagram
006502' 336 00 0 00 005503'             SKIPN   .SQARB+RCDARG           ;Any more datagrams?
006503' 254 00 0 00 006570'             JRST    INPDGX                  ;No, dismiss the interrupt
006504' 350 00 0 00 000257'             AOS     DGMSRV                  ;Count of recieved dgms
006505' 200 03 0 00 005502'             MOVE    AC3,.SQCID+RCDARG       ;Get the cid

                                ;Find the cblk that is connected to this incoming datagram.

006506' 476 00 0 00 000002              SETOM   AC2                     ;Init index pointer
006507' 201 01 0 00 000040              MOVEI   AC1,CBLSTE              ;Get size of cblst
006510' 350 00 0 00 000002      INPDGA: AOS     AC2                     ;Increment index
006511' 200 05 0 02 010556'             MOVE    AC5,CBLST(AC2)          ;Get cblk address
006512' 312 03 0 05 000000              CAME    AC3,CID(AC5)            ;Are the cids equal?
006513' 367 01 0 00 006510'             SOJG    AC1,INPDGA              ;No, keep looking
006514' 336 00 0 00 000001              SKIPN   AC1                     ;Have we looked in the whole table?
006515' 254 00 0 00 262467              CRASH   (NCI,No cblk for dgm input interrupt) ;No cblk for this dgm

                                ;The address of the received datagram cblk is now in ac5.
                                ;Check to see if this cblk is full(2 rhblk address).

006516' 332 00 0 05 000002              SKIPE   REQ1(AC5)                ;Is this location empty?
006517' 336 00 0 05 000003              SKIPN   REQ2(AC5)                ;No, is this location empty?
006520' 254 00 0 00 006527'             JRST    INPDGB                  ;Yes, we have room for this datagram

                                ;Cblk is full. Send back no resources.

006521' 200 01 0 00 005502'             MOVE    AC1,.SQCID+RCDARG       ;Get the cid
006522' 200 02 0 00 005503'             MOVE    AC2,.SQARB+RCDARG       ;Get the rblk address
006523' 476 00 0 00 000003              SETOM   AC3                     ;Was received as a datagram
006524' 260 17 0 00 005015'             GO      SENNRS                  ;Send back no resources
006525' 260 17 0 00 005370'             GO      QUDGM                   ;Queue up this same rblk
006526' 254 00 0 00 006501'             JRST    INPDGH                  ;See if any more datagrams are available
```

```
5080
5081                                    ;Find the rhblk from the rblk address
5082
5083  006527' 200 03 0 00 005503'   INPDGB: MOVE   AC3,.SQARB+RCDARG      ;Get the rblk address
5084  006530' 402 00 0 00 000001            SETZM  AC1                     ;Clr ac
5085  006531' 515 02 0 00 777400            HRLZI  AC2,-<COUNT*RHBPER>     ;Negative count of the no of rhblks
5086  006532' 316 03 0 01 010621'   INPDGC: CAMN   AC3,RH1+CTPPKT(AC1)     ;Does this rblk match?
5087  006533' 254 00 0 00 006537'           JRST   INPDGD                  ;Yes
5088  006534' 271 01 0 00 000004            ADDI   AC1,RHBLEN              ;Incr to next rhblk
5089  006535' 253 02 0 00 006532'           AOBJN  AC2,INPDGC              ;Have we checked all the rhblk's?
5090  006536' 254 00 0 00 262507            CRASH  (CFD,Can't find rhblk for dgm input interrupt)
5091
5092                                    ;Found the rhblk that is connected to this datagram.
5093                                    ;Set up the cblk as follows:
5094                                    ;       Cblk:   cid                    ;was here already
5095                                    ;               Status                 ;set the "GOBIT" and "DGM" in the status
5096                                    ;               Rhblk                  ;if this was 0 add the rhblk address
5097                                    ;               Rhblk                  ;or if this was 0 add the rhblk address
5098
5099  006537' 201 03 0 01 010616'   INPDGD: MOVEI  AC3,RH1(AC1)             ;Get the rhblk address
5100  006540' 332 00 0 05 000002            SKIPE  REQ1(AC5)               ;Is this location empty?
5101  006541' 254 00 0 00 006552'           JRST   INPDGE                  ;No, go ahead
5102  006542' 202 03 0 05 000002            MOVEM  AC3,REQ1(AC5)           ;Yes, store it in the cblk
5103  006543' 336 00 0 05 000003            SKIPN  REQ2(AC5)               ;Is this location empty?
5104  006544' 254 00 0 00 006561'           JRST   INPDGF                  ;Go ahead
5105  006545' 200 01 0 03 000002            MOVE   AC1,STATUS(AC3)         ;Get the status word
5106  006546' 661 01 0 00 000001            TLO    AC1,(GOBIT)             ;Set the gobit
5107  006547' 661 01 0 00 000020            TLO    AC1,(DGM)               ;Set the dgm bit
5108  006550' 202 01 0 03 000002            MOVEM  AC1,STATUS(AC3)         ; And store it
5109  006551' 254 00 0 00 006501'           JRST   INPDGH                  ;Don't put on linked list
5110
5111  006552' 336 00 0 05 000003   INPDGE: SKIPN  REQ2(AC5)               ;Is this location empty?
5112  006553' 202 03 0 05 000003            MOVEM  AC3,REQ2(AC5)           ;Yes, store it in the cblk
5113
5114                                    ;Set the gobit & dgm in  he rhblk.
5115
5116  006554' 200 01 0 03 000002            MOVE   AC1,STATUS(AC3)         ;Get the status word
5117  006555' 661 01 0 00 000001            TLO    AC1,(GOBIT)             ;Set the gobit
5118  006556' 661 01 0 00 000020            TLO    AC1,(DGM)               ;Set the dgm bit
5119  006557' 202 01 0 03 000002            MOVEM  AC1,STATUS(AC3)         ; And store it
5120  006560' 254 00 0 00 006501'           JRST   INPDGH                  ;See if any more datagrams are available
```

```
5121
5122                                    ;Set the gobit & dgm in the rhblk and add this rhblk to the linked list.
5123                                    ;Set up the rhblk as follows:
5124                                    ;       Rhblk:  flink                   ;set the flink up
5125                                    ;               Blink                   ;set the blink up
5126                                    ;               Dgm,gobit,,cblk         ;set the dgm & gobit. Cblk was already here.
5127                                    ;               Rblk                    ;rblk was already here
5128
5129  006561' 200 01 0 03 000002       INPDGF: MOVE    AC1,STATUS(AC3)         ;Get the status word
5130  006562' 661 01 0 00 000001              TLO     AC1,(GOBIT)             ;Set the gobit
5131  006563' 661 01 0 00 000020              TLO     AC1,(DGM)               ;Set the dgm bit
5132  006564' 202 01 0 03 000002              MOVEM   AC1,STATUS(AC3)         ; And store it
5133  006565' 200 01 0 00 000003              MOVE    AC1,AC3                 ;Rhblk in ac1
5134  006566' 260 17 0 00 003073'             GO      ADHBLK                  ;Add this rhblk to the linked list.
5135  006567' 254 00 0 00 006501'             JRST    INPDGH                  ;See if any more datagrams are available
5136
5137  006570' 260 17 0 00 004453'     INPDGX: GO      CWATSC                  ;Go check wait state
5138  006571' 332 00 0 00 000216'             TRMSG   (Exit from Input Datagram Interrupt)
5139  006572' 260 17 0 00 262522
5140  006573' 263 17 0 00 000000              RTN                             ;Return to dismiss interrupt
```

```
                                        SUBTTL  Interrupt Routine For Inputting Messages

                                ;

006574' 332 00 0 00 000216'     INPMSG: TRMSG   (Interrupt for Input of Message)
006575' 260 17 0 00 262536
006576' 260 17 0 00 005416'     INPMSH: GO      RCVMSG                  ;Receive a message
006577' 336 00 0 00 005447'             SKIPN   .SQARB+RCMARG           ;Any more messages?
006600' 254 00 0 00 006714'             JRST    INPMSX                  ;No, dismiss the interrupt
006601' 350 00 0 00 000260'             AOS     MSGSRV                  ;Count of recieved msgs
006602' 200 03 0 00 005446'             MOVE    AC3,.SQCID+RCMARG       ;Get the cid
                                ;
                                ;Find the cblk that is connected to this incoming message.
                                ;
006603' 402 00 0 00 000273'             SETZM   DGMMSG                  ;Clr the flag
006604' 476 00 0 00 000002              SETOM   AC2                     ;Init index pointer
006605' 201 01 0 00 000040              MOVEI   AC1,CBLSTE              ;Get size of cblst
006606' 350 00 0 00 000002      INPMSA: AOS     AC2                     ;Increment index
006607' 200 05 0 02 010556'             MOVE    AC5,CBLST(AC2)          ;Get cblk address
006610' 312 03 0 05 000000              CAME    AC3,CID(AC5)            ;Are the cids equal?
006611' 367 01 0 00 006606'             SOJG    AC1,INPMSA              ;No, keep looking
006612' 336 00 0 00 000001              SKIPN   AC1                     ;Have we looked in the whole table?
006613' 254 00 0 00 262557              CRASH   (CFM,Can't find cblk for message interrupt) ;No cblk for msg
```

```
5162
5163                                    ; he address of the received message cblk is now in ac5.
5164                                    ;Check to see if this cblk is full(2 rhblk address).
5165
5166  006614' 332 00 0 05 000002                SKIPE   REQ1(AC5)                ;Is this location empty?
5167  006615' 336 00 0 05 000003                SKIPN   REQ2(AC5)                ;No, is this location empty?
5168  006616' 254 00 0 00 006636'               JRST    INPMSB                   ;Yes, we have room for this message
5169
5170                                    ;We don't have any room for this incoming message.
5171                                    ;If this is a 3rd party response message set the no resource flag.
5172
5173  006617' 200 01 0 05 000001                MOVE    AC1,CBSTS(AC5)           ;Get the cblk status word
5174  006620' 607 01 0 00 200000                TLNN    AC1,(THIRD)              ;Is this a 3rd party cblk?
5175  006621' 254 00 0 00 006630'               JRST    INPMSJ                   ;No
5176  006622' 661 01 0 00 010000                TLO     AC1,(GANRES)             ;Set the "GEN ACT NO RESOURCE" flag
5177  006623' 202 01 0 05 000001                MOVEM   AC1,CBSTS(AC5)           ;Store it
5178  006624' 200 01 0 00 005446'               MOVE    AC1,.SQCID+RCMARG        ;Get the cid
5179  006625' 200 02 0 00 005447'               MOVE    AC2,.SQARB+RCMARG        ;Get the rblk address
5180  006626' 260 17 0 00 005342'               GO      QBMSG                    ;Queue up this same rblk
5181  006627' 254 00 0 00 006576'               JRST    INPMSH                   ;See if any more messages are available
5182
5183                                    ;Cblk is full. Send back no resources.
5184
5185  006630' 200 01 0 00 005446'  INPMSJ: MOVE    AC1,.SQCID+RCMARG        ;Get the cid
5186  006631' 200 02 0 00 005447'          MOVE    AC2,.SQARB+RCMARG        ;Get the rblk address
5187  006632' 402 00 0 00 000003           SETZM   AC3                      ;Was received as a message
5188  006633' 260 17 0 00 005015'          GO      SENNRS                   ;No, send back no resources
5189  006634' 260 17 0 00 005342'          GO      QBMSG                    ;Queue up this same rblk
5190  006635' 254 00 0 00 006576'          JRST    INPMSH                   ;See if any more messages are available
5191
5192                                    ;Find the rhblk from the rblk address
5193
5194  006636' 200 03 0 00 005447'  INPMSB: MOVE    AC3,.SQARB+RCMARG        ;Get the rblk address
5195  006637' 402 00 0 00 000001           SETZM   AC1                      ;Clr ac
5196  006640' 515 02 0 00 777400           HRLZI   AC2,-<COUNT*RHBPER>      ;Negative count of the no of rhblks
5197  006641' 316 03 0 01 010621'  INPMSC: CAMN    AC3,RH1+CTPPKT(AC1)      ;Does this rblk match?
5198  006642' 254 00 0 00 006646'          JRST    INPMSD                   ;Yes
5199  006643' 271 01 0 00 000004           ADDI    AC1,RHBLEN               ;Incr to next rhblk
5200  006644' 253 02 0 00 006641'          AOBJN   AC2,INPMSC               ;Have we checked all the rhblk's?
5201  006645' 254 00 0 00 262576           CRASH   (NRM,Can't find rhblk for message interrupt) ;No match
```

```
5202
5203                                    ;Found the rhblk that is connected to this message.
5204                                    ;Set up the cblk as follows:
5205                                    ;       Cblk:   cid                     ;was here already
5206                                    ;               Status                  ;set the "GOBIT" in the status
5207                                    ;               Rhblk                   ;if this was 0 add the rhblk address
5208                                    ;               Rhblk                   ;or if this was 0 add the rhblk address
5209
5210  006646' 201 03 0 01 010616'       INPMSD: MOVEI   AC3,RH1(AC1)            ;Get the rhblk address
5211  006647' 200 00 0 03 000003               MOVE    AC0,CTPPKT(AC3)         ;Get the address of ctpkt
5212  006650' 135 00 0 00 262601               LDB     AC0,[POINT 8,@AC0,7]    ;Get the opcode
5213  006651' 306 00 0 00 000005               CAIN    AC0,5                   ;Check if opcode=5
5214  006652' 476 00 0 00 000273'              SETOM   DGMMSG                  ;Yes, set the flag
5215  006653' 332 00 0 05 000002               SKIPE   REQ1(AC5)               ;Is this location empty?
5216  006654' 254 00 0 00 006674'              JRST    INPMSE                  ;No, go ahead
5217  006655' 202 03 0 05 000002               MOVEM   AC3,REQ1(AC5)           ;Yes, store it in the cblk
5218
5219                                    ;If a 3rd party connection set one of the generate activity continue flags.
5220
5221  006656' 200 01 0 05 000001               MOVE    AC1,CBSTS(AC5)          ;Get the cblk status word
5222  006657' 607 01 0 00 200000               TLNN    AC1,(THIRD)             ;Is this a 3rd party cblk?
5223  006660' 254 00 0 00 006664'              JRST    INPMSL                  ;No, continue
5224  006661' 661 01 0 00 004000               TLO     AC1,(GA1CON)            ;Set the gen act continue flag
5225  006662' 202 01 0 05 000001               MOVEM   AC1,CBSTS(AC5)          ;Store it
5226  006663' 254 00 0 00 006576'              JRST    INPMSH                  ;See if any more messages are available
5227
5228                                    ;Now check to see if we have another request(rhblk connected to this cblk.
5229                                    ;If we do, do not add this request(rhblk) to the linked list yet. It will
5230                                    ;Be added later when the 1st request in through via the "MDSNDC" subroutine.
5231
5232  006664' 332 00 0 05 000003       INPMSL: SKIPN   REQ2(AC5)               ;Is this location empty?
5233  006665' 254 00 0 00 006703'              JRST    INPMSF                  ;Yes, go ahead
5234  006666' 200 01 0 03 000002       INPMSM: MOVE    AC1,STATUS(AC3)         ;Get the status word
5235  006667' 661 01 0 00 000001               TLO     AC1,(GOBIT)             ;Set the gobit
5236  006670' 332 00 0 00 000273'              SKIPE   DGMMSG                  ;Skip if set
5237  006671' 661 01 0 00 000420               TLO     AC1,(DGM!DGMSG)         ;Set the dgm bit
5238  006672' 202 01 0 03 000002               MOVEM   AC1,STATUS(AC3)         ; And store it
5239  006673' 254 00 0 00 006576'              JRST    INPMSH                  ;Don't put on linked list
5240
5241  006674' 202 03 0 05 000003       INPMSE: MOVEM   AC3,REQ2(AC5)           ;Store rhblk address in the cblk
5242
5243                                    ;If a 3rd party connection set one of the generate activity continue flags.
5244
5245  006675' 607 01 0 00 200000               TLNN    AC1,(THIRD)             ;Is this a 3rd party cblk?
5246  006676' 254 00 0 00 006666'              JRST    INPMSM                  ;No, continue
5247  006677' 200 01 0 05 000001               MOVE    AC1,CBSTS(AC5)          ;Yes, get the cblk status word
5248  006700' 661 01 0 00 002000               TLO     AC1,(GA2CON)            ;Set the gen act continue flag
5249  006701' 202 01 0 05 000001               MOVEM   AC1,CBSTS(AC5)          ;Yes, then store it
5250  006702' 254 00 0 00 006576'              JRST    INPMSH                  ;See if any more messages are available
```

```
                                        ;Set the gobit in the rhblk and add this rhblk to the linked list.
                                        ;Set up the rhblk as follows:
                                        ;       Rhblk:  flink                   ;set the flink up
                                        ;               Blink                   ;set the blink up
                                        ;               Gobit,,cblk             ;set the gobit. Cblk was already here.
                                        ;               Rblk                    ;rblk was already here

006703' 200 01 0 03 000002      INPMSF: MOVE    AC1,STATUS(AC3)         ;Get the status word
006704' 661 01 0 00 000001              TLO     AC1,(GOBIT)             ;Set the gobit
006705' 332 00 0 00 000273'             SKIPE   DGMMSG                  ;Skip if set
006706' 661 01 0 00 000420              TLO     AC1,(DGM!DGMSG)         ;Set the dgm and dgmsg bit
006707' 202 01 0 03 000002              MOVEM   AC1,STATUS(AC3)         ; And store it
006710' 200 01 0 00 000003              MOVE    AC1,AC3                 ;Rhblk in ac1
006711' 402 00 0 00 000273'             SETZM   DGMMSG                  ;Clr the flag
006712' 260 17 0 00 003073'             GO      ADHBLK                  ;Add this rhblk to the linked list.
006713' 254 00 0 00 006576'             JRST    INPMSH                  ;See if any more messages are available

006714' 260 17 0 00 004453'     INPMSX: GO      CWATSC                  ;Go check wait state
006715' 332 00 0 00 000216'             TRMSG   (Exit Interrupt for Input of Message)
006716' 260 17 0 00 262612
006717' 263 17 0 00 000000              RTN                             ;Return to dismiss interrupt
```

```
                                        SUBTTL  Interrupt Routine For DMA Transfer Complete

006720' 332 00 0 00 000216'     DMAXFR: TRMSG   (Interrupt for Input DMA)
006721' 260 17 0 00 262625
006722' 260 17 0 00 006144'     DMAXFA: GO      DATAQ                   ;Get entry off data queue

                                ;Ac3 = buffer id whos xfer has completed

006723' 337 00 0 00 000003              SKIPG   AC3                     ;Any more data in queue?
006724' 254 00 0 00 006755'             JRST    DMAXFX                  ;No, dismiss interupt
006725' 350 00 0 00 000263'             AOS     DMACMP                  ;Count of dma xfers completed

006726' 476 00 0 00 000002              SETOM   AC2                     ;Init index pointer
006727' 201 01 0 00 000040              MOVEI   AC1,CBLSTE              ;Get size of cblst
006730' 350 00 0 00 000002      DMAXFB: AOS     AC2                     ;Increment index
006731' 200 04 0 02 010556'             MOVE    AC4,CBLST(AC2)          ;Get cblk address
006732' 316 03 0 04 000006              CAMN    AC3,BFSCA1(AC4)         ;Is this sca buffer the same?
006733' 254 00 0 00 006736'             JRST    DMAXFC                  ;Yes, set the dma xfer done bit
006734' 312 03 0 04 000010              CAME    AC3,BFSCA2(AC4)         ;Is this sca buffer the same?
006735' 254 00 0 00 006752'             JRST    DMAXFD                  ;No, continue
006736' 200 05 0 04 000001      DMAXFC: MOVE    AC5,CBSTS(AC4)          ;Get the cblk status
006737' 661 05 0 00 040000              TLO     AC5,(DMA)               ;Set dma completed flag
006740' 202 05 0 04 000001              MOVEM   AC5,CBSTS(AC4)          ;Store the cblk status

006741' 200 05 0 04 000002              MOVE    AC5,REQ1(AC4)           ;Get 1st RH header block address
006742' 336 00 0 00 000005              SKIPN   AC5                     ;Skip if this one isn't used
006743' 200 05 0 04 000003              MOVE    AC5,REQ2(AC4)           ;Get 2nd RH header block address
006744' 336 00 0 00 000005              SKIPN   AC5                     ;Skip if this one isn't used
006745' 254 04 0 00 000000              HALT                            ; Fatal error, no RH block in use
006746' 200 02 0 05 000002              MOVE    AC2,STATUS(AC5)         ;Get the status word
006747' 661 02 0 00 000001              TLO     AC2,(GOBIT)             ;Set the GO bit
006750' 202 02 0 05 000002              MOVEM   AC2,STATUS(AC5)         ;Update the status word

006751' 254 00 0 00 006754'             JRST    DMAXFS                  ;Get another event

006752' 367 01 0 00 006730'     DMAXFD: SOJG    AC1,DMAXFB              ;Look in all the cblks for a match

                                ;Could not find a cblk with this buffer name in any of the cblks.

006753' 254 00 0 00 262651              CRASH   (NCD,Can't find cblk for DMA transfer complete interrupt)

006754' 254 00 0 00 006722'     DMAXFS: JRST    DMAXFA                  ;Get another event from queue

006755' 332 00 0 00 000216'     DMAXFX: TRMSG   (Exit Input DMA Interrupt)
006756' 260 17 0 00 262662
006757' 260 17 0 00 004453'             GO      CWATSC                  ;Go check wait state
006760' 263 17 0 00 000000              RTN                             ;Return to dismiss interrupt
```

```
                                        SUBTTL  Interrupt Routine For Connections & Errors

006761' 332 00 0 00 000216'     OTHREV: TRMSG   (Connections and Errors Interrupt)
006762' 260 17 0 00 262676
006763' 260 17 0 00 006173'     OTHREA: GO      EVENTQ                  ;Get entry off event queue

                                ;Ac3=connect id  &  ac4=event code

006764' 337 00 0 00 000004              SKIPG   AC4                     ;Any more events in queue?
006765' 254 00 0 00 006770'             JRST    OTHREX                  ;No, dismiss interupt
006766' 260 17 1 04 006773'             GO      @EVTTBL(AC4)            ;Do the appropriate event
006767' 254 00 0 00 006763'             JRST    OTHREA                  ;Get another event from queue

006770' 332 00 0 00 000216'     OTHREX: TRMSG   (Exit Connection and Errors Interrupt)
006771' 260 17 0 00 262713
006772' 263 17 0 00 000000              RTN                             ;Return to dismiss interrupt

006773' 000 00 0 00 000000      EVTTBL: Z                               ;Indexing for this table starts at 1
006774' 000000  007010'                 VCBRKN                          ;1 - Virtual circuit broken
006775' 000000  007045'                 CNTOLS                          ;2 - Connect to listen
006776' 000000  007121'                 CONACC                          ;3 - Connection was accepted
006777' 000000  007131'                 CONREJ                          ;4 - Connection was rejected
007000' 000000  007134'                 MDSNDC                          ;5 - Msg/dgm send complete
007001' 000000  007175'                 LTLCRD                          ;6 - Little credit left
007002' 000000  007200'                 OFLINE                          ;7 - Node went offline
007003' 000000  007203'                 ONLINE                          ;8 - Node went online
007004' 000000  007206'                 OKSEND                          ;9 - Ck to send data
007005' 000000  007224'                 RMDISC                          ;10 - Remote initiated disconnect
007006' 000000  007227'                 PRTBRK                          ;11 - Port broke connection
007007' 000000  007232'                 CAVAIL                          ;12 - Credit available
```

```
                                ;********** Virtual circuit broken **********

                                ;Look for the cblk that has lost its virtual circuit.

007010' 332 00 0 00 000216'     VCBRKN: TRMSG   (<  Virtual circuit broken (1)>)
007011' 260 17 0 00 262727
007012' 476 00 0 00 000002              SETOM   AC2                     ;Init index pointer
007013' 201 01 0 00 000040              MOVEI   AC1,CBLSTE              ;Get size of cblst
007014' 350 00 0 00 000002      VCBRKA: AOS     AC2                     ;Increment index
007015' 200 06 0 02 010556'             MOVE    AC6,CBLST(AC2)          ;Get cblk address
007016' 312 03 0 06 000000              CAME    AC3,CID(AC6)            ;Are the cids equal?
007017' 367 01 0 00 007014'             SOJG    AC1,VCBRKA              ;No, keep looking
007020' 336 00 0 00 000001              SKIPN   AC1                     ;Have we looked in the whole table?
007021' 254 00 0 00 007044'             JRST    VCBRKC                  ;Yes, we don't have a cblk for this
                                                                        ;V.C. it must be another node gone away

                                ;The address of the lost virtual circuits cblk is now in ac6

007022' 550 01 0 00 000016              HRRZ    AC1,CURSTS              ;Get the cblk address of current rhblk
007023' 316 06 0 00 000001              CAMN    AC6,AC1                 ;Has the current cblks v. C. Gone?

                                ;Set the v. C. Gone bit in the current status ac(cursts).

007024' 661 16 0 00 000010              TLO     CURSTS,(VCGONE)         ;Yes, set v. C. Gone status bit

                                ;Set the v. C. Gone bit in the cblk.

007025' 200 01 0 06 000001              MOVE    AC1,CBSTS(AC6)          ;Get cblks status
007026' 661 01 0 00 000010              TLO     AC1,(VCGONE)            ;Set virtual circuit gone status bit
007027' 202 01 0 06 000001              MOVEM   AC1,CBSTS(AC6)          ;Now store it

                                ;Set the v. C. Gone bit in both of the rhblks.

007030' 550 05 0 06 000002              HRRZ    AC5,REQ1(AC6)           ;Get the 1st rhblk
007031' 336 00 0 00 000005              SKIPN   AC5                     ;Do we have a rhblk entry?
007032' 254 00 0 00 007036'             JRST    VCBRKB                  ;No, go ahead
007033' 200 01 0 05 000002              MOVE    AC1,STATUS(AC5)         ;Get rhblks status
007034' 661 01 0 00 000010              TLO     AC1,(VCGONE)            ;Set virtual circuit gone status bit
007035' 202 01 0 05 000002              MOVEM   AC1,STATUS(AC5)         ;Store rhblks status

007036' 550 05 0 06 000003      VCBRKB: HRRZ    AC5,REQ2(AC6)           ;Get the 2nd rhblk
007037' 336 00 0 00 000005              SKIPN   AC5                     ;Do we have a rhblk entry?
007040' 254 00 0 00 007044'             JRST    VCBRKC                  ;No, go ahead
007041' 200 01 0 05 000002              MOVE    AC1,STATUS(AC5)         ;Get rhblks status
007042' 661 01 0 00 000010              TLO     AC1,(VCGONE)            ;Set virtual circuit gone status bit
007043' 202 01 0 05 000002              MOVEM   AC1,STATUS(AC5)         ;Store rhblks status

                                ;A v.C. Has been closed, now go back to the mainline code instead of the
                                ;"WAIT" Jsys because if there aren't any more connections opened we want
                                ;To go to the routine "KIL3RD".

004453'                         VCBRKC: JRST    CWATSC                  ;Go check wait state and return
```

```
5403
5404                                    ;********** Connect to a listen **********
5405
5406  007045' 332 00 0 00 000216'       CNTOLS: TRMSG   (<  Connect to Listen (2)>)
5407  007046' 260 17 0 00 262742
5408
5409  007047' 312 03 0 00 000244'               CAME    AC3,CONIDC              ;Is this a connect from a cntrlr?
5410  007050' 254 00 0 00 007053'               JRST    CNTOLA                  ;No
5411  007051' 260 17 0 00 005136'               GO      LISTNC                  ;Yes, listen for another controller
5412  007052' 254 00 0 00 007057'               JRST    CNTOLB                  ;Go to common code
5413  007053' 312 03 0 00 000245'       CNTOLA: CAME    AC3,CONIDR              ;Is this a connect from a responder?
5414  007054' 254 00 0 00 007111'               JRST    CNTOLC                  ;No, must be garbage. Do a reject.
5415  007055' 260 17 0 00 005177'               GO      LISTNR                  ;Yes, listen for another responder
5416  007056' 476 00 0 00 000223'               SETOM   RESPFG                  ;Set responder flag
5417
5418                                    ;We get here only if we were connected to by either a controller or a
5419                                    ;responder.  Here we store the "CID" in the cblk and set the "GET INFORMATION"
5420                                    ;bit.
5421
5422  007057' 260 17 0 00 003144'       CNTOLB: GO      CBLKAV                  ;Get a connection block
5423  007060' 200 06 0 00 000001               MOVE    AC6,AC1                 ;Store the cblk in ac6
5424  007061' 201 02 0 00 000002               MOVEI   AC2,NRESRC              ;Reject reason=no resources
5425  007062' 336 00 0 00 000006               SKIPN   AC6                     ;Any cblks available?
5426  007063' 254 00 0 00 007112'               JRST    CNTOLD                  ;No, reject the connection
5427
5428  007064' 202 03 0 06 000000               MOVEM   AC3,CID(AC6)            ;Store the cid in the cblk
5429  007065' 402 00 0 00 000005               SETZM   AC5                     ;Clear ac
5430  007066' 661 05 0 00 100000               TLO     AC5,(GTINFO)            ;Set "GET INFORMATION" bits
5431  007067' 202 05 0 06 000001               MOVEM   AC5,CBSTS(AC6)          ;Store the cblk status
5432
5433                                    ;Queue 4 buffers for message reception.
5434
5435  007070' 402 00 0 00 000001               SETZM   AC1                     ;Tell the queue routine to find a rhblk
5436  007071' 200 02 0 00 000006               MOVE    AC2,AC6                 ;Get cblk address, cid already in ac3
5437  007072' 201 04 0 00 777777               MOVEI   AC4,-1                  ;0 = Queue dgrm, -1 = queue msg
5438  007073' 260 17 0 00 003604'               GO      QUEUE                   ;Queue a message buffer
5439  007074' 260 17 0 00 003604'               GO      QUEUE                   ;Queue a message buffer
5440  007075' 260 17 0 00 003604'               GO      QUEUE                   ;Queue a message buffer
5441  007076' 260 17 0 00 003604'               GO      QUEUE                   ;Queue a message buffer
5442  007077' 332 00 0 00 000223'               SKIPE   RESPFG                  ;Was this a connect from a responder?
5443  007100' 254 00 0 00 007115'               JRST    CNTOLE                  ;Yes, don't queue dgm buffers
5444
5445                                    ;Queue 4 buffers for datagram reception.
5446
5447  007101' 402 00 0 00 000001               SETZM   AC1                     ;Tell the queue routine to find a rhblk
5448  007102' 200 02 0 00 000006               MOVE    AC2,AC6                 ;Get cblk address, cid already in ac
5449  007103' 201 04 0 00 000000               MOVEI   AC4,0                   ;0 = Queue dgrm, -1 = queue msg
5450  007104' 260 17 0 00 003604'               GO      QUEUE                   ;Queue a datagram buffer
5451  007105' 260 17 0 00 003604'               GO      QUEUE                   ;Queue a datagram buffer
5452  007106' 260 17 0 00 003604'               GO      QUEUE                   ;Queue a datagram buffer
5453  007107' 260 17 0 00 003604'               GO      QUEUE                   ;Queue a datagram buffer
5454  007110' 254 00 0 00 007115'               JRST    CNTOLE                  ;Return
```

```
5455
5456  007111' 201 02 0 00 000000      CNTOLC: MOVEI   AC2,NMATCH              ;Reject reason=not a respdr or cntrlr
5457  007112' 200 01 0 00 000003      CNTOLD: MOVE    AC1,AC3                 ;Need the cid in ac1
5458  007113' 260 17 0 00 005266'             GO      REJECT                  ;Reject this connection
5459  007114' 254 00 0 00 007117'             JRST    CNTOLX                  ;Skip next code
5460
5461  007115' 200 01 0 06 000000      CNTOLE: MOVE    AC1,CID(AC6)            ;Need the cid in ac1
5462  007116' 260 17 0 00 005237'             GO      ACCEPT                  ;Accept the connection
5463
5464  007117' 402 00 0 00 000223'     CNTOLX: SETZM   RESPFG                  ;Clear responder flag
5465  007120' 263 17 0 00 000000              RTN                             ;Return
5466
5467                                  ;********** Connection was accepted(for 3rd party) **********
5468
5469  007121' 332 00 0 00 000216'     CONACC: TRMSG   (<  Connection was accepted (3)>)
5470  007122' 260 17 0 00 262756
5471
5472  007123' 260 17 0 00 003144'             GO      CBLKAV                  ;Get a connection block
5473  007124' 202 03 0 01 000000              MOVEM   AC3,CID(AC1)            ;Store the cid in the cblk
5474  007125' 402 00 0 00 000005              SETZM   AC5                     ;Clear ac
5475  007126' 661 05 0 00 300000              TLO     AC5,(THIRD!GTINFO)      ;Set 3rd party & get information bits
5476  007127' 202 05 0 01 000001              MOVEM   AC5,CBSTS(AC1)          ;Store the cblk status
5477  007130' 263 17 0 00 000000              RTN
5478
5479                                  ;********** Connection was rejected(for 3rd party) **********
5480
5481                                  ;Nothing to do here. There wasn't any cblks or rhblks assigned to this
5482                                  ;virtual circuit yet so there isn't any to deallocate. We basically just
5483                                  ;wait for the "EXWAIT" routine to time out, which will cause the test
5484                                  ;to send back a ctp response with "FUNCTION FAILURE" in the status byte.
5485
5486  007131' 332 00 0 00 000216'     CONREJ: TRMSG   (<  Connection was rejected (4)>)
5487  007132' 260 17 0 00 262772
5488  007133' 263 17 0 00 000000              RTN
```

```
                                ;********** Message or datagram send complete **********
                                ;We get interrupted here when a dgm or msg was sent across the ci by the
                                ;monitor.
007134' 332 00 0 00 000216'     MDSNDC: TRMSG   (<  Message datagram sent complete (5)>)
007135' 260 17 0 00 263007
007136' 476 00 0 00 000002              SETOM   AC2                     ;Init index pointer
007137' 201 01 0 00 000040              MOVEI   AC1,CBLSTE              ;Get size of cblst
007140' 350 00 0 00 000002      MDSND1: AOS     AC2                     ;Increment index
007141' 200 04 0 02 010556'             MOVE    AC4,CBLST(AC2)          ;Get cblk address
007142' 312 03 0 04 000000              CAME    AC3,CID(AC4)            ;Is this cid the same?
007143' 367 01 0 00 007140'             SOJG    AC1,MDSND1              ;No, look in all the cblks for a match
007144' 336 00 0 00 000001              SKIPN   AC1                     ;Did we find a match?
007145' 254 00 0 00 263030              CRASH   (CFC,Can't find cblk after send complete) ;No
                                ;We found the cblk that sent the msg or dgm.
007146' 336 01 0 04 000002              SKIPN   AC1,REQ1(AC4)           ;Is there an entry?
007147' 254 00 0 00 007161'             JRST    MDSND2                  ;No
007150' 200 05 0 01 000002              MOVE    AC5,STATUS(AC1)         ;Get its status word
007151' 607 05 0 00 000002              TLNN    AC5,(DONE)              ;Is "DONE" set in this rhblk?
007152' 254 00 0 00 007161'             JRST    MDSND2                  ;No
007153' 260 17 0 00 003112'             GO      RMHBLK                  ;Remove this rhblk from the linked list
                                ;Now requeue this buffer.  The rhblk is already in ac1 and the cid is already
                                ;in ac3.
007154' 200 02 0 00 000005              MOVE    AC2,AC5                 ;M = status,,cblk
007155' 260 17 0 00 003630'             GO      QUEBUF                  ;Go queue
007156' 336 01 0 04 000003              SKIPN   AC1,REQ2(AC4)           ;Is there another entry in the cblk?
007157' 254 00 0 00 007161'             JRST    MDSND2                  ;No
                                ;This is were we add the pending rhblk to the linked list.
007160' 260 17 0 00 003073'             GO      ADHBLK                  ;Add this rhblk to the linked list
007161' 336 01 0 04 000003      MDSND2: SKIPN   AC1,REQ2(AC4)           ;Is there an entry?
007162' 254 00 0 00 007174'             JRST    MDSND3                  ;No
007163' 200 05 0 01 000002              MOVE    AC5,STATUS(AC1)         ;Get its status word
007164' 607 05 0 00 000002              TLNN    AC5,(DONE)              ;Is "DONE" set in this rhblk?
007165' 254 00 0 00 007174'             JRST    MDSND3                  ;No
007166' 260 17 0 00 003112'             GO      RMHBLK                  ;Remove this rhblk from the linked list
```

```
5535
5536                                    ;Now requeue this buffer.  The rhblk is already in ac1 and the cid is alread
5537                                    ;in ac3.
5538
5539  007167' 200 02 0 00 000005                MOVE    AC2,AC5                 ;Status,,cblk to ac2
5540  007170' 260 17 0 00 003630'               GO      QUEBUF                  ;Go queue
5541
5542  007171' 336 01 0 04 000002                SKIPN   AC1,REQ1(AC4)           ;Is there another entry in the cblk?
5543  007172' 254 00 0 00 007174'               JRST    MDSND3                  ;No
5544
5545                                    ;This is were we add the pending rhblk to the linked list.
5546
5547  007173' 260 17 0 00 003073'               GO      ADHBLK                  ;Add this rhblk to the linked list
5548
5549  007174' 254 00 0 00 004453'     MDSND3: JRST    CWATSC                  ;Go check wait state and return
5550
5551                                    ;********** Little credit left **********
5552
5553                                    ;According to the ctp specification we only queue 6 buffers for each virtual
5554                                    ;Circuit, 2 for datagram and 4 for message reception. The cie's are only
5555                                    ;Going to send 1 request and then wait for a response before another is sent.
5556
5557  007175' 332 00 0 00 000216'     LTLCRD: TRMSG   (<  Little credit left (6)>)
5558  007176' 260 17 0 00 263041
5559  007177' 263 17 0 00 000000                RTN
5560
5561                                    ;********** Node went offline **********
5562
5563  007200' 332 00 0 00 000216'     OFLINE: TRMSG   (<  Node went offline (7)>)
5564  007201' 260 17 0 00 263054
5565  007202' 254 00 0 00 007010'               JRST    VCBRKN
5566
5567                                    ;********** Node went online **********
5568
5569  007203' 332 00 0 00 000216'     ONLINE: TRMSG   (<  Node went online (8)>)
5570  007204' 260 17 0 00 263066
5571  007205' 263 17 0 00 000000                RTN
```

```
5572
5573                                    ;********** Ok to send messages **********
5574
5575  007206' 332 00 0 00 000216'   OKSEND: TRMSG   (<  OK to send messages (9)>)
5576  007207' 260 17 0 00 263101
5577
5578  007210' 476 00 0 00 000002            SETOM   AC2                     ;Init index pointer
5579  007211' 201 01 0 00 000040            MOVEI   AC1,CBLSTE              ;Get size of cblst
5580  007212' 350 00 0 00 000002   OKSEN1:  AOS     AC2                     ;Increment index
5581  007213' 200 04 0 02 010556'           MOVE    AC4,CBLST(AC2)          ;Get cblk address
5582  007214' 312 03 0 04 000000            CAME    AC3,CID(AC4)            ;Is this cid the same?
5583  007215' 367 01 0 00 007212'           SOJG    AC1,OKSEN1              ;No, look in all the cblks for a match
5584  007216' 336 00 0 00 000001            SKIPN   AC1                     ;Did we find a match?
5585  007217' 254 00 0 00 263124            CRASH   (CCO,Can't find cblk on OK to send messages interrupt) ;No
5586
5587                                    ;We found the cblk that is ready to receive messages.
5588
5589  007220' 200 05 0 04 000001            MOVE    AC5,CBSTS(AC4)          ;Yes, get the cblk status
5590  007221' 661 05 0 00 020000            TLO     AC5,(OKTOSD)            ;Set "OK TO SEND" messages flag
5591  007222' 202 05 0 04 000001            MOVEM   AC5,CBSTS(AC4)          ;Store the cblk status
5592  007223' 263 17 0 00 000000            RTN
5593
5594                                    ;********** Remote initiated disconnect **********
5595
5596  007224' 332 00 0 00 000216'   RMDISC: TRMSG   (<  Remote initiated disconnect (10)>)
5597  007225' 260 17 0 00 263137
5598  007226' 254 00 0 00 007010'           JRST    VCBRKN
5599
5600                                    ;********** Port broke connection **********
5601
5602  007227' 332 00 0 00 000216'   PRTBRK: TRMSG   (<  Port broke connection (11)>)
5603  007230' 260 17 0 00 263153
5604  007231' 254 00 0 00 007010'           JRST    VCBRKN
5605
5606                                    ;********** Credit is available **********
5607
5608  007232' 332 00 0 00 000216'   CAVAIL: TRMSG   (<  Credit available (12)>)
5609  007233' 260 17 0 00 263166
5610  007234' 263 17 0 00 000000            RTN
```

```
                                        SUBTTL  Interrupt System Tables

                                ;TOPS-20 priority level and channel tables for the software interrupt system

                                ;Channel table - one word entry for each channel
                                ;All channels currently at priority level one.

                    000000              ONCHNS==0                       ;Activated channels
                    000000              CHNCNT==0                       ;Channel count
                                        DEFINE  ASSINT(CHAN,ADDR),<
                                        XWD     1,ADDR
                                        CHAN==CHNCNT
                                        CHNCNT==CHNCNT+1
                                        ONCHNS==ONCHNS!1B<CHAN>
                                >;end of DEFINE ASSINT

007235' 000001  004325'         CHNTAB: ASSINT  (TIMCHN,TIMR20)         ;Channel for timer interrupts
007236' 000001  006423'                 ASSINT  (DGMCHN,INPDG0)         ;Channel for datagram available
007237' 000001  006435'                 ASSINT  (MSGCHN,INPMS0)         ;Channel for message available
007240' 000001  006446'                 ASSINT  (DMACHN,DMAXF0)         ;Channel for dma xfer complete
007241' 000001  006457'                 ASSINT  (OTHCHN,OTHRE0)         ;Channel for connections & errors
007242'                                 BLOCK   ^D36-CHNCNT             ;Fill rest of table

                                ;Priority level table

007301' 000000  007304'         LEVTAB: PCLEV1                          ;Addr for pc stored for pl1
007302' 000000  007305'                 PCLEV2                          ;Addr for pc stored for pl2
007303' 000000  007306'                 PCLEV3                          ;Addr for pc stored for pl3

                                ;Storage for flags and pc

007304' 000 00 0 00 000000      PCLEV1: Z                               ;Pc interrupted for priority level 1
007305' 000 00 0 00 000000      PCLEV2: Z                               ;Pc interrupted for priority level 2
007306' 000 00 0 00 000000      PCLEV3: Z                               ;Pc interrupted for priority level 3

                                ;TOPS-10 interrupt system tables

                    000000      PIVECT: SCOFFS==.-PIVECT                ;Offset for SCS events
007307' 000000  006372'                 EXP     INTTEN                  ;New PC - for SCS events
007310' 000000  000000                  EXP     0                       ;Old PC
007311' 000000  777743                  XWD     0,,PCSCS                ;Interrupt type
007312' 000000  000000                  EXP     0                       ;Interrupt status
                    000004              TIOFFS==.-PIVECT                ;Offset for PITMR
007313' 000000  004313'                 EXP     TIMR10                  ;New PC
007314' 000000  000000                  EXP     0                       ;Old PC
007315' 000000  777776                  XWD     0,,PCTMR                ;Timer interrupts
007316' 000000  000000                  EXP     0                       ;Interrupt status
```

```
                        SUBTTL  Tables

                ;This table contains the number of bytes/response packet

                                                        ;Op -    name    - # of bytes
                                                        ;-------------------------
5664  007317' 000000  000046   NMBYTS: ^D38             ;64 - Function rsp - 38  mandatory
5665  007320' 000000  000024           ^D20             ;65 - Buff map rsp - 20
5666  007321' 000000  000024           ^D20             ;66 - Buf unmp rsp - 20
5667  007322' 000000  000040           ^D32             ;67 - Move buf rsp - 32
5668  007323' 000000  000200           ^D128            ;68 - Gen msg  rsp - 128 mandatory
5669  007324' 000000  001000           ^D512            ;69 - Gen dtgm rsp - 512 mandatory
5670  007325' 000000  000013           ^D11             ;70 - Gen rest rsp - 11
5671  007326' 000000  000020           ^D16             ;71 - Gen strt rsp - 16
5672  007327' 000000  000006           ^D6              ;72 - S/s unso rsp - 06  mandatory
5673  007330' 000000  000015           ^D13             ;73 - Con data rsp - 13  mandatory
5674  007331' 000000  000050           ^D40             ;74 - Cntr rd  rsp - 40  mandatory
5675  007332' 000000  000006           ^D6              ;75 - Connect  rsp - 06
5676  007333' 000000  000006           ^D6              ;76 - Lstr dis rsp - 13  mandatory
5677  007334' 000000  000024           ^D20             ;77 - Mbuf map rsp - 20
5678  007335' 000000  000024           ^D20             ;78 - Mbf umap rsp - 20
5679  007336' 000000  000034           ^D28             ;79 - Mov mbuf rsp - 28
5680  007337' 000000  000006           ^D6              ;80 - Mn state rsp - 06
5681  007340' 000000  001000           ^D512            ;Not implemented opcode
```

```
                                ;This spear table contains: Lh = total number of connections since the
                                ;responder was started.  Rh = number of connections started since the last
                                ;spear entry.  Port #0 is the 1st 16 entries of this table, port #1 is the next
                                ;16, and so on.  We can have a maximum of 6 ci's on a system and 16 nodes per
                                ;ci. 16X6 = 96 Since we can have a maximum of 8(0-7) ports on our system we
                                ;don't know what ports the 6 different ci's will be connected to, so we must
                                ;make room for all 8 ports. 16X8 = 128

007341' 000001  000001          SPEART: 1,,1
007342' 000002  000002                  2,,2
007343'                                 BLOCK   ^D126

                000200          SPEARE=.-SPEART                         ;Length of this block

                                ;This block is used for input to the "SYERR" jsys.

007541' 250000  000000          SYSERT: 250000,,0                       ;Event type
007542'                                 BLOCK   3                       ;3 Words of header info for syerr
007545' 00C000  020002          SYSE02: 20002                           ;Time of occurence block
007546'                                 BLOCK   1                       ;Internal date/time
007547' 000000  100005          SYSE10: 100005                          ;Diagnostic info block
007550' 44 46 43 51 43 00               SIXBIT/DFCIC/                   ;Diagnostic name
007551' 000000  000003                  DECVER                          ;Version number
007552' 777777  777777                  -1                              ;User proj,pgrm #(not used here)
007553' 777777  250000                  -1,,250000                      ;Unit#,,dev#,port#,mbz,node#
007554' 000000  120000          SYSE12: 120000                          ;Responder statistics block
007555'                                 BLOCK   1                       ;Time of last spear entry
007556'                                 BLOCK   ^D192                   ;2 Entries/node. 96X2 = 192

                000315          SYSERE=.-SYSERT                         ;Length of this block

                000013          SYSHLF=SYSE12-SYSERT                    ;Used for the header block length
```

```
                                ;The 10 words in the cblk are:
                                ;Cblkn: cid                     ;connection id of the ctp packet
                                ;       Status                  ;1b1(3rd party),1b2(get info),1b3(dma xfer),
                                ;                               ;  1B4(ok to send),1b5(ga nores),1b6(rcvd resp),
                                ;                               ;  1B7(rcvd resp),1b12(overdue),1b14(vcgone)
                                ;       1St req blk addr        ;address of rblk
                                ;       2Nd req blk addr        ;address of rblk
                                ;       Node #                  ;node #
                                ;       Local port #            ;references 1 of 6 possible ci's
                                ;       1St sca buffer          ;32 bit buffer name sca gives us
                                ;       1St bufx                ;our buffer address of above(--^) word
                                ;       2Nd sca buffer          ;32 bit buffer name sca gives us
                                ;       2Nd bufx                ;our buffer address of above(--^) word

                                DEFINE CBLK (N),<
                                        XLIST
                                CBLK'N: BLOCK   CBKLEN
                                        LIST
                                >
                        000001          N=1
                                        REPEAT  COUNT,<
                                        CBLK \N
                                        N=N+1
                                >

010544'
```

```
5743
5744                                    DEFINE CBLS (N),<
5745                                            XLIST
5746                                            CBLK'N
5747                                            LIST
5748                                    >
5749
5750                    000001                  N=1
5751                                    CBLST:  REPEAT  COUNT,<
5752                                            CBLS \N
5753                                            N=N+1
5754                                    >
5755
5756
5757                    000040          CBLSTE=.-CBLST
```

```
                                ;The request header block
                                ;
                                ; 6 Bit cid,,flink       ;points to the req hdr blk to be executed next
                                ; Repcount,,blink        ;points to the req hdr blk that was just executed
                                ; Status,,cblk           ;1b10(cont),1b11(waitt),1b12(overdue pkt),1b13(dgm),
                                ;                        ; 1B14(vcgone),1b15(onetim),1b16(done),1b17(gobit),,cblkx address
                                ; Ctppkt                 ;the address(rbx) of the ctp packet
                                ;
                                DEFINE RHBLK (N),<
                                        XLIST
                                RH'N:   BLOCK   <RHBLEN-1>
                                        RB'N
                                        LIST
                                >
                        000001          N=1
                                        REPEAT  <COUNT*RHBPER>,<
                                        RHBLK \N
                                        N=N+1
                                >
                                ;
                                ;Storage for timer routines.
                                ;
012616'                         DLYADR: BLOCK   COUNT           ;Address of rhblk for this timer request
012656'                         DLYUPT: BLOCK   COUNT           ;Uptime at which to process this request
012716'                         DLYRTN: BLOCK   COUNT           ;Routine to call when this alarm goes off
```

```
5787
5788                            ;This is the storage area for ctp packets. We use the following buffers for
5789                            ;queueing messages & datagrams. When we execute an scs% receive message or
5790                            ;datagram the contents go here.
5791
5792                            ;This next "LOC" statement forces the assembler to an even address page
5793                            ;boundary.  The 140 octal makes up for the "JOB DATA" area the link loader adds
5794                            ;to the front of this program.
5795
5796                            ;LOC <.+140+777>&777000
5797  033000                    LOC <.+20000+777>&777000        ;Room for DDT
5798
5799                            ;The scs% jsys will not transfer data across page boundries.  Therefore the
5800                            ;following macro was created to insure that no buffer(rbx) will cross a page
5801                            ;boundary even if the size of some of the parameters change.
5802
5803                            ;M=^d512/mxdgm                  ;m = the # of req blocks that can fit on a page
5804                            ;N=1                            ;number to start the request blocks(eg. Rb1)
5805                            ;Repeat <<<count*rhbper>+m-1>/m>,<      ;round out the # of repeats
5806                            ;Loc <.+777>&777000             ;Start at an even page boundary
5807                            ;Repeat <m>,<                   ;generate n buffers/page
5808                            ;Rblk \n                        ;macro to generate request block buffers
5809                            ;N=n+1                          ;bump number
5810
5811                            DEFINE RBLK (N),<
5812                            RB'N:   BLOCK   <MXDGM>
5813                            >
5814  000003                            M=^D512/MXDGM
5815  000001                            N=1
5816                                    REPEAT  <<<COUNT*RHBPER>+M-1>/M>,<
5817                                    LOC <.+777>&777000
5818                                    REPEAT  <M>,<
5819                                    RBLK \N
5820                                    N=N+1
5821                           >>
5822  160460
```

```
                                ;This loc statement gets us to an even address boundary.
                                
161000                          LOC <.+777>&777000
                                
                                ;1000 Octal is the maximum number of words(1 page) allowed.
                                
                                DEFINE BUF (N),<
                                BUF'N:  BLOCK   ^D512
                                >
                                
                        000001          N=1
                                        REPEAT  <COUNT*2>,<
                                        BUF \N
                                        N=N+1
                                >
260000
                                
                                ;Make pointers to buffer addresses
                                
                                DEFINE BUFLS (N),<
                                        XLIST
                                        BUF'N
                                        LIST
                                >
                                
                        000001          N=1
                                BUFLST: REPEAT  COUNT,<
                                        BUFLS \N
                                        N=N+1
                                >
                                
                                
                                
                        000040  BUFSTE=.-BUFLST
```

```
                                    ;Literals
                                    
261040                              LIT..:  XLIST                       ;Literal label
                                            LIST
                                    
                    000362'                 END     START
```

NO ERRORS DETECTED

PROGRAM BREAK IS 012756
ABSOLUTE BREAK IS 263201
CPU TIME USED 00:26.168

121P CORE USED

AC0	000000	spd	ACT06E	002050'		BUF11	171000		BUF72	252000			
AC1	000001	spd	ACT06F	002070'		BUF12	172000		BUF73	253000			
AC10	000010	spd	ACT06H	002072'		BUF13	173000		BUF74	254000			
AC11	000011	spd	ACT06J	002076'		BUF14	174000		BUF75	255000			
AC12	000012	spd	ACT06S	002104'		BUF15	175000		BUF76	256000			
AC13	000013	spd	ACT06X	002102'		BUF16	176000		BUF77	257000			
AC16	000016	spd	ACT07	002203'		BUF17	177000		BUFAA	003251'			
AC2	000002	spd	ACT07A	002221'		BUF2	162000		BUFARG	005704'			
AC3	000003	spd	ACT07B	002231'		BUF20	200000		BUFAV	003243'			
AC4	000004	spd	ACT07C	002233'		BUF21	201000		BUFD	000351'			
AC5	000005	spd	ACT07D	002245'		BUF22	202000		BUFF1	000007	spd		
AC6	000006	spd	ACT07E	002250'		BUF23	203000		BUFF2	000011	spd		
AC7	000007	spd	ACT07F	002274'		BUF24	204000		BUFFSZ	005661'			
ACCARG	005263'		ACT07G	002270'		BUF25	205000		BUFL	000340'			
ACCEPT	005237'		ACT07H	002276'		BUF26	206000		BUFLEN	000227'			
ACT00	000560'		ACT07J	002302'		BUF27	207000		BUFLIM	000372	spd		
ACT01	000701'		ACT07S	002310'		BUF3	163000		BUFLST	261000			
ACT01A	000715'		ACT07X	002306'		BUF30	210000		BUFNAM	000230'			
ACT01B	000775'		ACT08	002342'		BUF31	211000		BUFO	000347'			
ACT01C	001011'		ACT08A	002376'		BUF32	212000		BUFS	000345'			
ACT01H	001003'		ACT08C	002367'		BUF33	213000		BUFSTE	000040			
ACT01L	000732'		ACT08S	002401'		BUF34	214000		BUFT	000315'			
ACT01M	000740'		ACT08X	002377'		BUF35	215000		CONR	000350'			
ACT01N	000742'		ACT09	002447'		BUF36	216000		C1NR	000355'			
ACT01S	001024'		ACT09A	002505'		BUF37	217000		CANTI1	004305'			
ACT01X	001022'		ACT09S	002507'		BUF4	164000		CANTIM	004301'			
ACT02	001111'		ACT10	002554'		BUF40	220000		CAVAIL	007232'			
ACT02A	001126'		ACT10A	002704'		BUF41	221000		CBKLEN	000012	spd		
ACT02B	001136'		ACT10B	002706'		BUF42	222000		CBLK1	010056'			
ACT02C	001146'		ACT10S	002710'		BUF43	223000		CBLK10	010164'			
ACT02E	001151'		ACT10X	002707'		BUF44	224000		CBLK11	010176'			
ACT02S	001157'		ACT11	003002'		BUF45	225000		CBLK12	010210'			
ACT02X	001160'		ACT11B	003020'		BUF46	226000		CBLK13	010222'			
ACT03	001322'		ACT11S	003022'		BUF47	227000		CBLK14	010234'			
ACT03A	001335'		ACT11X	003023'		BUF5	165000		CBLK15	010246'			
ACT03B	001355'		ACT12	003046'		BUF50	230000		CBLK16	010260'			
ACT03C	001365'		ACT12X	003053'		BUF51	231000		CBLK17	010272'			
ACT03D	001367'		ACT3B1	001363'		BUF52	232000		CBLK2	010070'			
ACT03E	001401'		ACTB	000325'		BUF53	233000		CBLK20	010304'			
ACT03F	001404'		ACTC	000320'		BUF54	234000		CBLK21	010316'			
ACT03G	001415'		ACTN	000343'		BUF55	235000		CBLK22	010330'			
ACT03I	001417'		ADHBLA	003106'		BUF56	236000		CBLK23	010342'			
ACT03J	001455'		ADHBLK	003073'		BUF57	237000		CBLK24	010354'			
ACT03S	001467'		AIC%	104000	000131	int	BUF6	166000		CBLK25	010366'		
ACT03T	001470'		AICARG	004420'			BUF60	240000		CBLK26	010400'		
ACT03X	001464'		AICLEN	000005			BUF61	241000		CBLK27	010412'		
ACT03Z	001466'		ANNOUN	000000'			BUF62	242000		CBLK3	010102'		
ACT04	001700'		BFSCA1	000006	spd		BUF63	243000		CBLK30	010424'		
ACT04S	001712'		BFSCA2	000010	spd		BUF64	244000		CBLK31	010436'		
ACT04X	001713'		BLINK	000001	spd		BUF65	245000		CBLK32	010450'		
ACT06	002006'		BOTEXE	000275'			BUF66	246000		CBLK33	010462'		
ACT06A	002024'		BUD0	000353'			BUF67	247000		CBLK34	010474'		
ACT06B	002031'		BUF1	161000			BUF7	167000		CBLK35	010506'		
ACT06C	002033'		BUF10	170000			BUF70	250000		CBLK36	010520'		
ACT06D	002045'		BUF100	260000			BUF71	251000		CBLK37	010532'		

```
CBLK4           010114'          CTPCOX          002750'          DMARD           000262'          EXDLY           004170'
CBLK40          010544'          CTPPKT          000003  spd      DMAWRT          000261'          EXDLY1          004212'
CBLK5           010126'          CUREXE          000276'          DMAXF0          006446'          EXDLY2          004215'
CBLK6           010140'          CURSTS          000016  spd      DMAXFA          006722'          EXDLY3          004216'
CBLK7           010152'          CWAT1           004460'          DMAXFB          006730'          EXDLY4          004217'
CBLKAA          003152'          CWAT11          004467'          DMAXFC          006736'          EXECT           000502'
CBLKAV          003144'          CWATSC          004453'          DMAXFD          006752'          EXIT    047000  000012
CBLST           010556'          CYCL60          000235'          DMAXFR          006720'          EXTE            000322'
CBLSTE          000040           DATAQ           006144'          DMAXFS          006754'          EXWAI1          004240'
CBSTS           000001  spd      DATARG          006170'          DMAXFX          006755'          EXWAI3          004243'
CHGOPC          003543'          DATATE          006161'          DONE    000002  000000  spd      EXWA[illegible]          004244'
CHGOPX          003556'          DATET1          004543'          DOSCS           004605'          EXWA[illegible]          004225'
CHKVCA          003660'          DATETM          004537'          DOSCS1          004623'          FAKEE[illegible]         000002  spd
CHKVCB          003664'          DEBRK%  104000  000136  int      DOSCS2          004617'          FCMD            000326'
CHKVCS          003650'          DEBRK.  047000  000137           DSDG            000356'          FLINK           000000  spd
CHKVCT          003677'          DECVER          000003  spd      DUM1            005620'          FSTS            000344'
CHKVCX          003700'          DELA            000316'          EIR%    104000  000126  int      FUNCFL          000373  spd
CHNCNT          000005  spd      DGM     000020  000000  spd      ENABL1          004503'          FUNCST          000537'
CHNTAB          007235'          DGMCHN          000001  spd      ENABLE          004472'          GA1CON  004000  000000  spd
CID             000000  spd      DGMMSG          000273'          ENAIN1          004446'          GA2CON  002000  000000  spd
CIDLOC          000246'          DGMSG   000400  000000  spd      ENAIN2          004451'          GAFLAG          000264'
CIPORT          000256'          DGMSRV          000257'          ENAINT          004440'          GANRES  010000  000000  spd
CNOA            000341'          DIAAJ%          000007  spd      ENDF            000276'  spd     GENC            000327'
CNON            000346'          DIABA%          000017  spd      ENTER           004033'          GENCON          000225'
CN1A            000352'          DIACI%          000020  spd      ENTER   077000  000000           GENDGM          001477'
CN1N            000354'          DIAFC%          000010  spd      ENTERC          004044'          GENF            000323'
CNFARG          005767'          DIAG%   104000  000530  int      ENTERD          004046'          GENFUN          000224'
CNFDA1          002422'          DIAG.   047000  000163           ENTERE          004051'          GENL            000336'
CNFDAT          002410'          DIAGAG          006342'          ENTERG          004053'          GENLEN          000226'
CNTLR           000301'          DIAGF           006264'          ENTERO          004060'          GENMS1          001514'
CNTOLA          000353'          DIAGJ           006247'          ENTERP          004076'          GENMS2          001540'
CNTOLB          007057'          DIAGS           006263'          ENTERS          004066'          GENMS4          001543'
CNTOLC          007111'          DIAGU           006261'          ENTREN          004055'          GENMS5          001545'
CNTOLD          007112'          DIAIA%          000002  spd      EPCAP%  104000  000151  int      GENMS6          001523'
CNTOLE          007115'          DIAIF%          000013  spd      ERCAL   320740  000000  int      GENMSG          001477'
CNTOLS          007045'          DIANC%          000015  spd      ERJMP   320700  000000  int      GENRS1          001734'
CNTOLX          007117'          DIANP%          000001  spd      ERRDI1          006275'          GENRS2          001750'
CNTRLR          000277'          DIANR%          000016  spd      ERRDI2          006271'          GENRS3          001752'
CNTRR1          002523'          DIATO%          000021  spd      ERRDI3          006306'          GENRST          001717'
CNTRRD          002515'          DIAVC%          000014  spd      ERRDI4          006313'          GENST1          002131'
CONACC          007121'          DIR%    104000  000130  int      ERRDIL          000013  spd      GENST2          002145'
CONARG          005126'          DISARG          005337'          ERRDIT          006326'          GENST3          002147'
CONDAT          000311'          DISCON          005314'          ERRSC1          004641'          GENSTR          002114'
CONECT          005055'          DISIN1          004433'          ERRSC2          004647'          GET     262740  000000
CONFIG          005737'          DISIN2          004436'          ERRSC4          004654'          GETACT          003364'
CONFIX          005760'          DISINT          004425'          ERRSCL          000035  spd      GETARG          006033'
CONIDC          000244'          DLYADR          012616'          ERRSCS          004633'          GETBUA          006027'
CONIDR          000245'          DLYGO           004352'          ERRSCT          004712'          GETBUF          006013'
CONREJ          007131'          DLYRBK          000243'          ERSTR%  104000  000011  int      GETCID          003536'
CONT    000200  000000  spd      DLYRTN          012716'          ERSTRT          004676'          GETER%  104000  000012  int
COUNT           000040  spd      DLYTIM          000242'          EVENTE          006214'          GETGLN          003416'
CTPCOA          002726'          DLYUPT          012656'          EVENTQ          006173'          GETINA          003720'
CTPCOB          002733'          DMA     040000  000000  spd      EVENTX          006236'          GETINB          003773'
CTPCOC          002746'          DMACHN          000003  spd      EVTARG          006237'          GETIND          003760'
CTPCON          002714'          DMACMP          000263'          EVTTBL          006773'          GETINE          003771'
```

```
GETINF          003706'          LDSTUS          003450'          NOP     600000  000000  sin      PRINTA          004706'
GETINX          003774'          LEVTAB          007301'          NORESO          000375  spd      PRINTB          004710'
GETREP          003364'          LICARG          005172'          NOTIMP          003337'          PRINTC          004547'
GETTAB  047000  000041           LIRARG          005232'          NOTMSG          005010'          PRINTD          004564'
GETTI2          004527'          LISDI1          003044'          NOUT%   104000  000224  int      PRINTO          004555'
GO      260740  000000           LISDIS          003027'          NRESRC          000002  spd      PRINTT          004550'
GOBIT   000001  000000  spd      LISTNC          005136'          NRSARG          005050'          PRTBRK          007227'
GPRS            003562'          LISTNR          005177'          NTIMPL          000377  spd      PS.FAC  004000  000000  spd
GPRS1           003576'          LIT..           261040           OFLINE          007200'          PS.FOF  200000  000000  spd
GPRSB           003601'          LK.LLC          000010  spd      OKSEN1          007212'          PS.FON  100000  000000  spd
GTAD%   104000  000227  int      LK.LLS          000001  spd      OKSEND          007206'          PS.TMS  400000  000000  spd
GTINFO  100000  000000  spd      LK.LNE          000002  spd      OKTOSD  020000  000000  spd      PSOUT%  104000  000076  int
HALT    254200  000000           LK.LNP          000004  spd      ONCHNS  760000  000000  spd      PUT     261740  000000
HALTF%  104000  000170  int      LOCALN          005635'          ONETIM  000004  000000  spd      QBDARG          005413'
HALTP           004671'          LOCARG          005657'          ONLINE          007203'          QBDGM           005370'
HIBER   047000  000072           LOCK    047000  000060           OPCARY          000577'          QBMARG          005365'
IFPIN1          004402'          LOCPRT          000005  spd      OPCHK           003521'          QBMSG           005342'
IFPIN5          004413'          LTLCRD          007175'          OPCO            000313'          QUEBUF          003630'
IFPINT          004360'          M               000003           OPCODE          000231'          QUEBUX          003642'
INPDGO          006424'          MAINA           000447'          OTHCHN          000004  spd      QUEUE           003604'
INPDGA          006510'          MAINC           000463'          OTHE            000331'          RB1             033000
INPDGB          006527'          MAINC1          000472'          OTHERN          000232'          RB10            035230
INPDGC          006532'          MAINLN          000437'          OTHREO          006457'          RB100           060000
INPDGD          006537'          MAPBU1          000622'          OTHREA          006763'          RB101           060230
INPDGE          006552'          MAPBUF          000607'          OTHREV          006761'          RB102           060460
INPDGF          006561'          MATCH           000001  spd      OTHREX          006770'          RB103           061000
INPDGH          006501'          MAXDGM          000266'          OURNOD          000247'          RB104           061230
INPDGM          006477'          MAXMSG          000265'          OUTCHR  051040  000000           RB105           061460
INPDGX          006570'          MAXOPC          000020  spd      OUTSTR  051140  000000           RB106           062000
INPMSO          006435'          MDSND1          007140'          OVRDUE  000040  000000  spd      RB107           062230
INPMSA          006606'          MDSND2          007161'          P               000017  spd      RB11            035460
INPMSB          006636'          MDSND3          007174'          POST            000335'          RB110           062460
INPMSC          006641'          MDSNDC          007134'          P10T11          003502'          RB111           063000
INPMSD          006646'          MONTYP          000234'          P11T10          003463'          RB112           063230
INPMSE          006674'          MOVE            000324'          P1ST            000337'          RB113           063460
INPMSF          006703'          MOVEB1          001201'          PCLEV1          007304'          RB114           064000
INPMSG          006574'          MOVEB2          001223'          PCLEV2          007305'          RB115           064230
INPMSH          006576'          MOVEB4          001230'          PCLEV3          007306'          RB116           064460
INPMSJ          006630'          MOVEB5          001231'          PCLREV          000000  spd      RB117           065000
INPMSL          006664'          MOVEBA          001206'          PCLTYP          000000  spd      RB12            036000
INPMSM          006666'          MOVEBF          001165'          PCLVER          000003  spd      RB120           065230
INPMSX          006714'          MOVETY          000233'          PIINI.  047000  000135           RB121           065460
INTDMS          006467'          MSGCHN          000002  spd      PISYS.  047000  000136           RB122           066000
INTFLG          000220'          MSGDGM          005550'          PITMR.  047000  000203           RB123           066230
INTTEN          006372'          MSGSRV          000260'          PIVECT          007307'          RB124           066460
KIL3RA          003272'          MXDGM           000230  spd      PKTIM           000334'          RB125           067000
KIL3RB          003301'          N               000041           PKTSZF          000367  spd      RB126           067230
KIL3RC          003304'          NMATCH          000000  spd      PKTZ            000333'          RB127           067460
KIL3RD          003263'          NMBYTS          007317'          PLIST           000014'          RB13            036230
KIL3RE          003330'          NOAC            000332'          PLISTE          000215'          RB130           070000
KIL3RF          003312'          NOBUFM          000371  spd      PLISTS          000015'          RB131           070230
KIL3RH          003325'          NOCONN          000366  spd      PORT            000330'          RB132           070460
KIL3RX          003331'          NODE            000004  spd      PRINT1          004573'          RB133           071000
LASTF           000222'          NODEUN          000370  spd      PRINT2          004574'          RB134           071230
LBUFOF          000271'          NONSEN          000376  spd      PRINT3          004575'          RB135           071460
```

RB136	072000	RB216	112000	RB277	132230	RB357	152230
RB137	072230	RB217	112230	RB3	033460	RB36	044460
RB14	036460	RB22	040460	RB30	042460	RB360	152460
RB140	072460	RB220	112460	RB300	132460	RB361	153000
RB141	073000	RB221	113000	RB301	133000	RB362	153230
RB142	073230	RB222	113230	RB302	133230	RB363	153460
RB143	073460	RB223	113460	RB303	133460	RB364	154000
RB144	074000	RB224	114000	RB304	134000	RB365	154230
RB145	074230	RB225	114230	RB305	134230	RB366	154460
RB146	074460	RB226	114460	RB306	134460	RB367	155000
RB147	075000	RB227	115000	RB307	135000	RB37	045000
RB15	037000	RB23	041000	RB31	043000	RB370	155230
RB150	075230	RB230	115230	RB310	135230	RB371	155460
RB151	075460	RB231	115460	RB311	135460	RB372	156000
RB152	076000	RB232	116000	RB312	136000	RB373	156230
RB153	076230	RB233	116230	RB313	136230	RB374	156460
RB154	076460	RB234	116460	RB314	136460	RB375	157000
RB155	077000	RB235	117000	RB315	137000	RB376	157230
RB156	077230	RB236	117230	RB316	137230	RB377	157460
RB157	077460	RB237	117460	RB317	137460	RB4	034000
RB16	037230	RB24	041230	RB32	043230	RB40	045230
RB160	100000	RB240	120000	RB320	140000	RB400	160000
RB161	100230	RB241	120230	RB321	140230	RB401	160230
RB162	100460	RB242	120460	RB322	140460	RB402	160460
RB163	101000	RB243	121000	RB323	141000	RB41	045460
RB164	101230	RB244	121230	RB324	141230	RB42	046000
RB165	101460	RB245	121460	RB325	141460	RB43	046230
RB166	102000	RB246	122000	RB326	142000	RB44	046460
RB167	102230	RB247	122230	RB327	142230	RB45	047000
RB17	037460	RB25	041460	RB33	043460	RB46	047230
RB170	102460	RB250	122460	RB330	142460	RB47	047460
RB171	103000	RB251	123000	RB331	143000	RB5	034230
RB172	103230	RB252	123230	RB332	143230	RB50	050000
RB173	103460	RB253	123460	RB333	143460	RB51	050230
RB174	104000	RB254	124000	RB334	144000	RB52	050460
RB175	104230	RB255	124230	RB335	144230	RB53	051000
RB176	104460	RB256	124460	RB336	144460	RB54	051230
RB177	105000	RB257	125000	RB337	145000	RB55	051460
RB2	033230	RB26	042000	RB34	044000	RB56	052000
RB20	040000	RB260	125230	RB340	145230	RB57	052230
RB200	105230	RB261	125460	RB341	145460	RB6	034460
RB201	105460	RB262	126000	RB342	146000	RB60	052460
RB202	106000	RB263	126230	RB343	146230	RB61	053000
RB203	106230	RB264	126460	RB344	146460	RB62	053230
RB204	106460	RB265	127000	RB345	147000	RB63	053460
RB205	107000	RB266	127230	RB346	147230	RB64	054000
RB206	107230	RB267	127460	RB347	147460	RB65	054230
RB207	107460	RB27	042230	RB35	044230	RB66	054460
RB21	040230	RB270	130000	RB350	150000	RB67	055000
RB210	110000	RB271	130230	RB351	150230	RB7	035000
RB211	110230	RB272	130460	RB352	150460	RB70	055230
RB212	110460	RB273	131000	RB353	151000	RB71	055460
RB213	111000	RB274	131230	RB354	151230	RB72	056000
RB214	111230	RB275	131460	RB355	151460	RB73	056230
RB215	111460	RB276	132000	RB356	152000	RB74	056460

RB75		057000		RH134	011372'	RH214	011672'	RH275	012176'
RB76		057230		RH135	011376'	RH215	011676'	RH276	012202'
RB77		057460		RH136	011402'	RH216	011702'	RH277	012206'
RBUFOF		000272'		RH137	011406'	RH217	011706'	RH3	010626'
RCDARG		005501'		RH14	010672'	RH22	010722'	RH30	010752'
RCMARG		005445'		RH140	011412'	RH220	011712'	RH300	012212'
RCVDGE		005471'		RH141	011416'	RH221	011716'	RH301	012216'
RCVDGM		005452'		RH142	011422'	RH222	011722'	RH302	012222'
RCVMSE		005435'		RH143	011426'	RH223	011726'	RH303	012226'
RCVMSG		005416'		RH144	011432'	RH224	011732'	RH304	012232'
REALBU		000001	spd	RH145	011436'	RH225	011736'	RH305	012236'
REDARG		006137'		RH146	011442'	RH226	011742'	RH306	012242'
REDATA		006115'		RH147	011446'	RH227	011746'	RH307	012246'
REFE		000314'		RH15	010676'	RH23	010726'	RH31	010756'
REJARG		005311'		RH150	011452'	RH230	011752'	RH310	012252'
REJECT		005266'		RH151	011456'	RH231	011756'	RH311	012256'
REPC		000321'		RH152	011462'	RH232	011762'	RH312	012262'
REQ1		000002	spd	RH153	011466'	RH233	011766'	RH313	012266'
REQ2		000003	spd	RH154	011472'	RH234	011772'	RH314	012272'
RESET%	104000	000147	int	RH155	011476'	RH235	011776'	RH315	012276'
RESOFS		000100	spd	RH156	011502'	RH236	012002'	RH316	012302'
RESPFG		000223'		RH157	011506'	RH237	012006'	RH317	012306'
RH1		010616'		RH16	010702'	RH24	010732'	RH32	010762'
RH10		010652'		RH160	011512'	RH240	012012'	RH320	012312'
RH100		011212'		RH161	011516'	RH241	012016'	RH321	012316'
RH101		011216'		RH162	011522'	RH242	012022'	RH322	012322'
RH102		011222'		RH163	011526'	RH243	012026'	RH323	012326'
RH103		011226'		RH164	011532'	RH244	012032'	RH324	012332'
RH104		011232'		RH165	011536'	RH245	012036'	RH325	012336'
RH105		011236'		RH166	011542'	RH246	012042'	RH326	012342'
RH106		011242'		RH167	011546'	RH247	012046'	RH327	012346'
RH107		011246'		RH17	010706'	RH25	010736'	RH33	010766'
RH11		010656'		RH170	011552'	RH250	012052'	RH330	012352'
RH110		011252'		RH171	011556'	RH251	012056'	RH331	012356'
RH111		011256'		RH172	011562'	RH252	012062'	RH332	012362'
RH112		011262'		RH173	011566'	RH253	012066'	RH333	012366'
RH113		011266'		RH174	011572'	RH254	012072'	RH334	012372'
RH114		011272'		RH175	011576'	RH255	012076'	RH335	012376'
RH115		011276'		RH176	011602'	RH256	012102'	RH336	012402'
RH116		011302'		RH177	011606'	RH257	012106'	RH337	012406'
RH117		011306'		RH2	010622'	RH260	010742'	RH34	010772'
RH12		010662'		RH20	010712'	RH260	012112'	RH340	012412'
RH120		011312'		RH200	011612'	RH261	012116'	RH341	012416'
RH121		011316'		RH201	011616'	RH262	012122'	RH342	012422'
RH122		011322'		RH202	011622'	RH263	012126'	RH343	012426'
RH123		011326'		RH203	011626'	RH264	012132'	RH344	012432'
RH124		011332'		RH204	011632'	RH265	012136'	RH345	012436'
RH125		011336'		RH205	011636'	RH266	012142'	RH346	012442'
RH126		011342'		RH206	011642'	RH267	012146'	RH347	012446'
RH127		011346'		RH207	011646'	RH27	010746'	RH35	010776'
RH13		010666'		RH21	010716'	RH270	012152'	RH350	012452'
RH130		011352'		RH210	011652'	RH271	012156'	RH351	012456'
RH131		011356'		RH211	011656'	RH272	012162'	RH352	012462'
RH132		011362'		RH212	011662'	RH273	012166'	RH353	012466'
RH133		011366'		RH213	011666'	RH274	012172'	RH354	012472'

```
RH355   012476'  RH75    011176'  SCSERC  000237'  SSQBF%  000021  spd
RH356   012502'  RH76    011202'  SCSFCN  000240'  SSRCF%  000016  spd
RH357   012506'  RH77    011206'  SEDARG  006110'  SSRQE%  000014  spd
RH36    011002'  RHBLEN  000004  spd SEDATA  006066'  SSSCP%  000033  spd
RH360   012512'  RHBLKA  003062'  SEN04   004147'  SSSTL%  000034  spd
RH361   012516'  RHBLKB  003067'  SEN04C  004163'  SSTMS%  000031  spd
RH362   012522'  RHBLKV  003056'  SEND    004124'  SSUNS1  002332'
RH363   012526'  RHBPER  000010  spd SENDA   004142'  SSUNS0  002320'
RH364   012532'  RMCBLA  003174'  SENDMA  005536'  SSVNO%  000012  spd
RH365   012536'  RMCBLB  003230'  SENDMC  005543'  STAARG  005733'
RH366   012542'  RMCBLC  003233'  SENDMD  005506'  STAR    000342'
RH367   012546'  RMCBLG  003212'  SENIMA  005001'  START   000362'
RH37    011006'  RMCBLH  003221'  SENIMP  004750'  STAT    000317'
RH370   012552'  RMCBLK  003164'  SENNRA  005041'  STATSI  005707'
RH371   012556'  RMCBLM  003224'  SENNRS  005015'  STATUS  000002  spd
RH372   012562'  RMDISC  007224'  SETTI1  004275'  STPARG  005626'
RH373   012566'  RMHBLK  003112'  SETTI2  004273'  STPOL1  005606'
RH374   012572'  RMRBLA  003134'  SETTI6  004255'  STPOL2  005613'
RH375   012576'  RMRBLB  003137'  SETTI7  004251'  STPOLL  005555'
RH376   012602'  RMVARG  006064'  SETTIM  004250'  STRACT  003402'
RH377   012606'  RMVBUA  006054'  SIR%  104000  000125  int  STRGLN  003434'
RH4     010632'  RMVBUF  006040'  SIXCID  000000  spd  STRREP  003402'
RH40    011012'  RPCAP%  104000  000150  int SPEAR   004005'  STRTAD  000252'
RH400   012612'  RPCNT   000001  spd SPEARA  004023'  STRTF   000217'
RH41    011016'  RQREFU  000374  spd SPEARE  000200  SUCCES  000000  spd
RH42    011022'  RSPDR   000305'  SPEARF  000217'  SYERR%  104000  000527  int
RH43    011026'  RSPNDR  000300'  SPEART  007341'  SYSAC2  000255'
RH44    011032'  RTN     263740  000000  SPEARX  004026'  SYSE02  007545'
RH45    011036'  S..CAI   004407'  spd SPOFST  000020  spd  SYSE10  007547'
RH46    011042'  S..CCO   007217'  spd SPRBLN  000357'  SYSE12  007554'
RH47    011046'  S..CFC   007145'  spd SPRCIN  000360'  SYSERE  000315'
RH5     010636'  S..CFD   006536'  spd SPRNOD  000361'  SYSERT  007541'
RH50    011052'  S..CFM   006613'  spd SQ%CVD  000010  sin  SYSHLF  000013
RH51    011056'  S..CII   004404'  spd SSACR%  000003  spd  SYSJYS  004111'
RH52    011062'  S..CLJ   004505'  spd SSACS%  000004  spd  THIRD  200000  000000  spd
RH53    011066'  S..CST   004412'  spd SSARG%  000002  spd  TIMCHN  000000  spd
RH54    011072'  S..DBF   004254'  spd SSCBF%  000022  spd  TIME%  104000  000014  int
RH55    011076'  S..EON   004450'  spd SSCPN%  000005  spd  TIMELN  000074  spd
RH56    011102'  S..NCD   006753'  spd SSCRB%  000026  spd  TIMER%  104000  000522  int
RH57    011106'  S..NCI   006515'  spd SSCUB%  000027  spd  TIMESP  000254'
RH6     010642'  S..NRH   003066'  spd SSDCF%  000017  spd  TIMIN1  004341'
RH60    011112'  S..NRM   006645'  spd SSDQE%  000024  spd  TIMIN7  004347'
RH61    011116'  S..POF   004435'  spd SSEQE%  000025  spd  TIMINT  004337'
RH62    011122'  S..RNQ   003077'  spd SSICI%  000013  spd  TIMR10  004313'
RH63    011126'  SC%DGA  200000  000000  spd SSIDM%  000032  spd  TIMR20  004325'
RH64    011132'  SC%DTA  100000  000000  spd SSIFC%  000001  spd  TIOFFS  000004  spd
RH65    011136'  SC%EVA  040000  000000  spd SSINN%  000010  spd  TOPEXE  000274'
RH66    011142'  SC%MSA  400000  000000  spd SSNBQ%  000015  spd  TRACE   000216'
RH67    011146'  SCALBF   000267'  SSNFB%  000020  spd  TSTTBL  000515'
RH7     010646'  SCARBF   000270'  SSNFC%  000011  spd  UNMAP1  001047'
RH70    011152'  SCOFFS   000000  spd SSNNK%  000007  spd  UNMAPB  001034'
RH71    011156'  SCS%    104000  000622  int SSNPC%  000006  spd  UNSACT  000253'
RH72    011162'  SCS.    047000  000213  SSNPV%  000000  spd  UTIM1   004515'
RH73    011166'  SCSBLK   000241'  SSNSB%  000030  spd  UTIME   004507'
RH74    011172'  SCSCAL   000236'  SSPSF%  000023  spd  VAL00   000553'
```

VAL01	000625'	VCGONE	000010	000000	spd	.SQCDA	000002	spd	.SSSTS	000012	spd
VAL01A	000645'	VEDIT		000014	spd	.SQCDT	000004	spd	.SSUMP	000015	spd
VAL01C	000654'	WAIT%	104000	000306	int	.SQCID	000001	spd	.TIMEL	000001	sin
VAL01D	000662'	WAITF		000221'		.SQDFL	000003	spd			
VAL01G	000641'	WAITIM		000250'		.SQDIS	000002	spd			
VAL01S	000673'	WAITRB		000251'		.SQDPN	000002	spd			
VAL01X	000671'	WAITT	000100	000000	spd	.SQESB	000002	spd			
VAL02	001052'	WATOVR		004355'		.SQEVT	000003	spd			
VAL02A	001072'	%CNDTM	000053	000011	spd	.SQFLG	000004	spd			
VAL02G	001066'	%CNMNT	000112	000011	spd	.SQFST	000002	spd			
VAL02S	001103'	%CNSTS	000017	000011	spd	.SQLCI	000004	spd			
VAL02X	001101'	%CNSUP	000136	000011	spd	.SQLDG	000002	spd			
VAL03	001236'	.DGCTR		000107	sin	.SQLEN	000000	spd			
VAL03A	001254'	.DGENB		000111	sin	.SQLMG	000001	spd			
VAL03B	001261'	.DGRST		000105	sin	.SQLNN	000001	spd			
VAL03C	001265'	.DGSTR		000106	sin	.SQLPN	000023	spd			
VAL03F	001275'	.FHSLF		400000	sin	.SQLPT	000003	spd			
VAL03G	001302'	.LBACC		000003	spd	.SQLRP	000004	spd			
VAL03H	001304'	.LBCON		000010	spd	.SQNAM	000001	spd			
VAL03S	001314'	.LBCSP		000007	spd	.SQOFS	000004	spd			
VAL03X	001312'	.LBDIS		000003	spd	.SQOSB	000002	spd			
VAL04	001552'	.LBEVT		000010	spd	.SQRCI	000007	spd			
VAL04A	001570'	.LBGDE		000003	spd	.SQREJ	000002	spd			
VAL04B	001577'	.LBGLN		000002	spd	.SQRNM	000003	spd			
VAL04C	001601'	.LBLIS		000005	spd	.SQRPN	000001	spd			
VAL04D	001603'	.LBQRD		000003	spd	.SQRPS	000002	spd			
VAL04E	001632'	.LBQRM		000003	spd	.SQSAD	000004	spd			
VAL04F	001640'	.LBRBS		000003	spd	.SQSBI	000005	spd			
VAL04G	001646'	.LBRCD		000024	spd	.SQSBR	000003	spd			
VAL04H	001656'	.LBRDG		000005	spd	.SQSNM	000002	spd			
VAL04S	001670'	.LBREJ		000003	spd	.SQSPN	000001	spd			
VAL04X	001666'	.LBREQ		000005	spd	.SQSYS	000003	spd			
VAL06	001757'	.LBRMG		000005	spd	.SQVCS	000003	spd			
VAL06A	001772'	.LBRPS		000003	spd	.SSACC	000023	spd			
VAL06S	002001'	.LBSMG		000005	spd	.SSAIC	000020	spd			
VAL06X	001777'	.LBSND		000005	spd	.SSCON	000000	spd			
VAL07	002154'	.LBSTS		000004	spd	.SSCSP	000010	spd			
VAL07A	002167'	.LBUMP		000002	spd	.SSDIS	000003	spd			
VAL07S	002176'	.PCSCS	777777	777743	spd	.SSEVT	000025	spd			
VAL07X	002174'	.PCTMR	777777	777776	spd	.SSGDE	000024	spd			
VAL08	002335'	.PRIOU		000101	sin	.SSGLN	000030	spd			
VAL09	002425'	.PSVIS		000003	spd	.SSLIS	000001	spd			
VAL09S	002442'	.PSVOP		000001	spd	.SSMAP	000014	spd			
VAL09X	002440'	.SIDGA		000000	spd	.SSQRD	000005	spd			
VAL10	002532'	.SIDMA		000002	spd	.SSQRM	000007	spd			
VAL10S	002547'	.SIMSA		000001	spd	.SSRBS	000035	spd			
VAL10X	002545'	.SIPAN		000003	spd	.SSRCD	000011	spd			
VAL11	002753'	.SQADC		000006	spd	.SSRDG	000022	spd			
VAL11A	002767'	.SQAFB		000002	spd	.SSREJ	000002	spd			
VAL11S	002775'	.SQAMC		000005	spd	.SSREQ	000017	spd			
VAL11X	002773'	.SQAPT		000002	spd	.SSRMG	000013	spd			
VCBRKA	007014'	.SQARB		000002	spd	.SSRPS	000036	spd			
VCBRKB	007036'	.SQBDN		000004	spd	.SSSDG	000004	spd			
VCBRKC	007044'	.SQBID		000002	spd	.SSSMG	000006	spd			
VCBRKN	007010'	.SQBNA		000002	spd	.SSSND	000016	spd			

SEQ 0226

AC0	44#	362	363	383	384	385	410	411	415	421	422	423	424	425
	426	435	437	456	457	459	463	464	468	469	470	472	563	564
	566	571	576	577	578	580	582	584	632	639	640	642	644	646
	650	651	653	662	668	670	671	673	681	685	698	728	733	734
	739	746	747	751	752	757	759	765	766	772	774	776	777	785
	786	787	788	791	792	798	800	812	835	842	843	845	847	849
	856	858	860	867	871	881	883	884	892	895	896	899	904	905
	910	919	926	936	962	964	976	981	991	992	994	996	1002	1003
	1010	1011	1017	1019	1020	1022	1028	1030	1032	1039	1044	1054	1060	1069
	1083	1084	1088	1089	1094	1096	1097	1105	1106	1112	1131	1138	1139	1141
	1142	1145	1151	1159	1161	1162	1165	1167	1170	1172	1179	1180	1214	1223
	1248	1250	1263	1268	1277	1278	1280	1286	1289	1293	1296	1303	1312	1313
	1317	1318	1323	1334	1335	1341	1343	1345	1346	1354	1355	1356	1357	1360
	1361	1367	1369	1377	1378	1382	1394	1398	1404	1410	1420	1430	1447	1458
	1463	1469	1470	1472	1478	1479	1481	1488	1494	1513	1518	1519	1523	1541
	1557	1559	1567	1574	1577	1589	1599	1616	1627	1632	1638	1639	1641	1647
	1648	1650	1657	1663	1682	1687	1688	1693	1694	1698	1716	1732	1734	1739
	1740	1742	1749	1756	1759	1771	1794	1795	1797	1803	1808	1809	1813	1814
	1815	1838	1846	1869	1875	1876	1877	1879	1886	1890	1898	1910	1917	1927
	1934	1940	1963	1969	1970	1971	1973	1980	1985	1991	2020	2022	2023	2025
	2028	2029	2031	2034	2035	2037	2040	2041	2043	2046	2047	2049	2052	2053
	2055	2057	2058	2105	2107	2112	2122	2127	2132	2150	2154	2160	2161	2162
	2164	2169	2176	2181	2194	2196	2201	2209	2211	2220	2270	2287	2288	2295
	2306	2318	2320	2324	2326	2346	2387	2447	2486	2492	2500	2515	2537	2542
	2548	2553	2554	2555	2556	2557	2567	2582	2587	2608	2609	2610	2625	2630
	2651	2652	2653	2665	2671	2674	2695	2696	2697	2698	2699	2700	2701	2702
	2721	2722	2723	2724	2725	2726	2727	2728	2776	2781	2782	2784	2785	2786
	2789	2798	2806	2870	2884	2915	2917	2938	2961	2962	2971	2972	2976	2977
	2978	2979	3003	3004	3021	3232	3240	3241	3242	3249	3255	3265	3277	3280
	3288	3294	3301	3322	3323	3330	3338	3339	3509	3512	3513	3514	3517	4315
	4316	4608	4609	4610	4620	4962	4987	4993	5003	5013	5023	5040	5211	5212
	5213													
AC1	45#	278	281	284	285	286	289	290	293	294	295	296	297	300
	301	302	305	306	309	310	313	314	315	316	317	318	319	322
	323	326	327	330	331	334	335	338	341	348	350	352	367	368
	369	370	371	372	374	378	409	410	420	421	424	430	434	435
	436	442	443	462	463	467	468	494	495	508	510	511	514	515
	517	549	555	556	572	573	574	575	577	579	580	581	582	583
	617	625	626	633	636	637	638	649	658	659	660	667	680	686
	695	696	703	707	715	716	718	720	722	724	726	736	737	738
	743	744	745	756	811	819	827	828	836	839	840	841	855	866
	872	878	879	880	916	925	935	937	948	949	958	959	960	962
	975	982	985	986	990	1001	1009	1016	1027	1053	1061	1074	1075	1076
	1080	1081	1082	1093	1124	1125	1127	1128	1156	1157	1158	1164	1169	1174
	1175	1213	1222	1224	1236	1237	1244	1245	1246	1248	1262	1269	1274	1275
	1276	1309	1310	1311	1374	1375	1376	1393	1399	1419	1429	1431	1444	1445
	1447	1451	1452	1457	1464	1466	1467	1468	1477	1487	1495	1503	1504	1505
	1517	1534	1535	1537	1538	1546	1547	1548	1549	1550	1554	1555	1556	1562
	1588	1598	1600	1613	1614	1616	1620	1621	1626	1633	1635	1636	1637	1646
	1656	1664	1672	1673	1674	1686	1692	1709	1710	1712	1713	1721	1722	1723
	1724	1725	1729	1730	1731	1744	1770	1779	1786	1787	1804	1811	1812	1820
	1821	1823	1830	1831	1833	1845	1854	1861	1862	1870	1872	1873	1874	1885
	1891	1901	1906	1907	1921	1931	1939	1948	1957	1958	1964	1966	1967	1968

SEQ 0227

	1979	1986	1996	1997	1998	2002	2003	2004	2005	2006	2010	2016	2017	2062
	2063	2064	2065	2066	2067	2068	2071	2077	2078	2079	2080	2081	2085	2091
	2092	2093	2094	2095	2096	2097	2098	2099	2100	2111	2121	2145	2146	2155
	2157	2158	2159	2167	2175	2182	2184	2191	2200	2208	2215	2216	2225	2228
	2231	2254	2258	2271	2276	2277	2282	2286	2288	2292	2294	2307	2313	2318
	2319	2324	2325	2330	2331	2333	2339	2340	2345	2361	2369	2388	2399	2400
	2405	2410	2411	2412	2417	2418	2424	2425	2430	2431	2436	2439	2446	2462
	2470	2487	2497	2498	2520	2521	2526	2532	2535	2541	2549	2552	2563	2564
	2566	2580	2583	2584	2585	2586	2588	2589	2590	2592	2602	2604	2605	2606
	2607	2612	2623	2626	2627	2628	2629	2631	2632	2633	2635	2645	2647	2648
	2649	2650	2655	2666	2668	2669	2670	2673	2740	2743	2751	2763	2764	2765
	2766	2777	2780	2788	2797	2800	2801	2802	2808	2835	2840	2842	2843	2847
	2848	2858	2871	2883	2893	2898	2925	2939	2952	2953	2958	2960	2961	2966
	2970	2971	2977	2979	2983	2984	2985	2989	3000	3002	3006	3007	3008	3011
	3020	3041	3046	3051	3060	3065	3085	3088	3101	3108	3109	3114	3123	3124
	3132	3139	3141	3146	3176	3180	3193	3203	3207	3218	3233	3237	3238	3239
	3245	3264	3278	3281	3300	3314	3320	3323	3324	3331	3347	3350	3352	3353
	3354	3357	3362	3372	3377	3387	3396	3401	3419	3425	3430	3431	3432	3438
	3439	3440	3441	3442	3443	3444	3445	3446	3451	3464	3468	3472	3473	3474
	3476	3481	3484	3488	3489	3490	3492	3501	3502	3504	3514	3524	3532	3533
	3534	3552	3553	3554	3555	3556	3560	3561	3562	3563	3564	3565	3566	3576
	3577	3578	3584	3595	3602	3612	3618	3619	3620	3640	3641	3642	3643	3654
	3663	3664	3665	3666	3669	3677	3686	3689	3694	3699	3703	3719	3728	3731
	3777	3794	3796	3803	3807	3828	3834	3842	3844	3850	3855	3883	3885	3894
	3895	3899	3900	3904	3905	3911	3915	3917	3942	3946	3947	3955	3957	3961
	3962	3964	3968	3987	3991	3992	3999	4001	4005	4006	4008	4013	4033	4039
	4042	4047	4052	4071	4077	4079	4085	4090	4108	4114	4116	4122	4127	4148
	4154	4156	4162	4166	4186	4192	4194	4200	4205	4222	4226	4227	4232	4240
	4245	4264	4268	4269	4274	4282	4287	4311	4317	4322	4324	4348	4354	4373
	4379	4382	4388	4392	4393	4395	4396	4398	4399	4403	4405	4412	4434	4441
	4447	4452	4468	4476	4482	4487	4504	4510	4513	4519	4524	4544	4550	4553
	4554	4555	4556	4563	4568	4612	4614	4618	4623	4643	4649	4650	4659	4662
	4686	4690	4697	4727	4731	4738	4760	4763	4764	4767	4776	4779	4794	4797
	4798	4801	4802	4803	4804	4814	4821	4824	4831	4854	4858	4865	4872	4877
	4884	4885	4889	4893	4894	4896	4897	4900	4905	4963	4969	4971	4972	4974
	4976	4978	4986	4994	5004	5014	5024	5039	5047	5057	5061	5062	5063	5074
	5084	5086	5088	5090	5099	5105	5106	5107	5108	5116	5117	5118	5119	5129
	5130	5131	5132	5133	5139	5144	5155	5159	5160	5161	5173	5174	5176	5177
	5178	5185	5195	5197	5199	5201	5210	5221	5222	5224	5225	5234	5235	5237
	5238	5245	5247	5248	5249	5259	5260	5262	5263	5264	5271	5276	5286	5308
	5312	5317	5323	5334	5356	5358	5362	5363	5369	5370	5378	5379	5380	5387
	5388	5389	5394	5395	5396	5407	5423	5435	5447	5457	5461	5470	5473	5476
	5487	5496	5499	5503	5504	5505	5509	5511	5522	5529	5531	5542	5558	5564
	5570	5576	5579	5583	5584	5585	5597	5603	5609					
AC10	52#													
AC11	53#													
AC12	54#													
AC13	55#													
AC16	56#	58												
AC2	46#	431	509	513	516	634	679	687	704	773	774	784	786	788
	799	800	810	837	865	873	888	890	891	924	938	947	948	957
	958	974	983	1038	1039	1052	1062	1101	1103	1104	1118	1120	1121	1185
	1195	1212	1225	1235	1236	1243	1244	1261	1270	1275	1276	1302	1303	1310

SEQ 0228

	1311	1331	1332	1342	1343	1353	1355	1357	1368	1369	1375	1376	1392	1400
	1418	1432	1443	1444	1456	1465	1467	1468	1486	1496	1512	1513	1528	1530
	1531	1555	1556	1587	1601	1612	1613	1625	1634	1636	1637	1655	1665	1681
	1682	1703	1705	1706	1730	1731	1769	1805	1810	1811	1844	1871	1873	1874
	1884	1892	1902	1917	1918	1927	1928	1938	1965	1967	1968	1978	1987	2017
	2018	2023	2027	2029	2033	2035	2039	2041	2045	2047	2051	2053	2056	2058
	2110	2156	2158	2159	2174	2183	2185	2190	2191	2199	2226	2227	2230	2242
	2245	2249	2260	2272	2281	2286	2287	2292	2293	2308	2330	2331	2332	2333
	2334	2338	2341	2344	2358	2362	2364	2365	2375	2389	2393	2395	2396	2445
	2459	2463	2465	2466	2476	2488	2508	2512	2527	2540	2550	2551	2552	2565
	2581	2584	2585	2587	2588	2590	2591	2603	2605	2606	2607	2608	2610	2611
	2624	2627	2628	2630	2631	2633	2634	2646	2648	2649	2650	2651	2653	2654
	2667	2669	2670	2672	2691	2693	2694	2702	2704	2717	2719	2720	2728	2730
	2744	2745	2746	2747	2748	2749	2778	2779	2780	2787	2796	2803	2809	2836
	2843	2847	2849	2857	2872	2876	2878	2882	2894	2899	2901	2902	2924	2940
	2967	3001	3012	3019	3042	3047	3048	3051	3054	3059	3066	3076	3085	3088
	3091	3092	3093	3098	3100	3102	3103	3104	3122	3123	3124	3125	3131	3140
	3142	3145	3177	3182	3192	3204	3209	3217	3234	3238	3239	3263	3279	3299
	3318	3319	3330	3348	3351	3352	3356	3363	3371	3378	3386	3392	3394	3396
	3398	3399	3400	3420	3426	3428	3433	3450	3527	3544	3546	3549	3557	3620
	3623	3625	3626	3656	3663	3664	3665	3683	3687	3688	3690	3691	3692	3697
	3701	3705	3720	3721	3778	3781	3782	3783	3784	3786	3787	3791	3792	3797
	3806	3829	3835	3839	3840	3845	3854	3875	3878	3879	3880	3881	3882	3886
	3887	3895	3896	3897	3900	3901	3902	3905	3906	3907	3912	3920	3943	3948
	3949	3950	3951	3952	3953	3958	3967	3988	3993	3994	3995	3996	3997	4002
	4012	4034	4037	4038	4040	4041	4043	4051	4072	4078	4080	4089	4109	4115
	4117	4126	4149	4155	4157	4165	4187	4193	4195	4204	4223	4233	4244	4265
	4275	4286	4312	4318	4326	4353	4374	4380	4381	4383	4411	4435	4442	4451
	4469	4477	4486	4505	4514	4523	4545	4551	4557	4567	4611	4615	4644	4647
	4648	4651	4658	4687	4691	4728	4732	4761	4768	4775	4795	4805	4813	4822
	4826	4830	4855	4871	4887	4891	4895	4902	4907	4908	4964	4969	4973	4975
	4977	4979	4980	4985	4995	5005	5015	5025	5038	5056	5058	5059	5075	5085
	5089	5154	5156	5157	5179	5186	5196	5200	5285	5287	5288	5302	5303	5304
	5357	5359	5360	5424	5436	5448	5456	5498	5500	5501	5519	5539	5578	5580
	5581													
AC3	47#	635	637	638	649	667	678	688	737	738	744	745	756	771
	775	782	790	797	801	809	838	840	841	855	864	874	879	880
	889	900	901	923	984	986	990	1001	1009	1016	1027	1051	1063	1081
	1082	1093	1102	1108	1119	1130	1157	1158	1164	1169	1186	1196	1211	1271
	1340	1344	1351	1359	1366	1370	1391	1477	1497	1517	1529	1540	1586	1646
	1666	1686	1692	1704	1715	1768	1806	1843	1893	1905	1916	1919	1922	1926
	1929	1932	1937	2243	2244	2246	2248	2258	2259	2309	2319	2320	2325	2326
	2343	2359	2363	2367	2368	2374	2390	2394	2441	2444	2460	2464	2468	2469
	2475	2489	2491	2493	2494	2507	2509	2510	2511	2514	2516	2517	2539	2692
	2694	2695	2697	2699	2701	2703	2718	2720	2721	2723	2725	2727	2729	2741
	2742	2743	2750	2837	2848	2856	2873	2881	2895	2900	2905	2906	2923	2941
	2947	3013	3018	3043	3058	3067	3077	3078	3081	3094	3130	3178	3181	3182
	3186	3191	3205	3208	3209	3210	3211	3216	3235	3262	3332	3364	3370	3379
	3385	3406	3407	3412	3413	3421	3449	3527	3603	3613	3621	3622	3688	3689
	3722	3779	3785	3786	3788	3789	3790	3791	3805	3830	3843	3853	3876	3897
	3901	3902	3906	3919	3944	3966	3989	4011	4035	4050	4073	4088	4110	4125
	4150	4164	4188	4203	4224	4243	4266	4285	4313	4319	4329	4330	4331	4342
	4343	4352	4375	4410	4436	4450	4470	4485	4506	4522	4546	4566	4645	4657

	4688	4729	4771	4773	4808	4823	4829	4856	4870	4894	4895	4965	4984	4996
	5006	5016	5026	5037	5052	5060	5076	5083	5086	5099	5102	5105	5108	5112
	5116	5119	5129	5132	5133	5149	5158	5187	5194	5197	5210	5211	5217	5234
	5238	5241	5259	5263	5264	5281	5289	5291	5361	5409	5413	5428	5457	5473
	5502	5582												
AC4	48#	689	719	720	725	726	764	773	784	799	808	891	892	894
	895	899	903	904	1064	1104	1105	1106	1121	1122	1127	1127	1138	1142
	1187	1188	1197	1198	1210	1272	1330	1331	1332	1342	1351	1368	1390	1498
	1531	1532	1534	1537	1585	1667	1706	1707	1709	1712	1767	1807	1842	1894
	1907	1921	1931	1936	2360	2365	2366	2369	2373	2391	2396	2397	2399	2405
	2410	2412	2417	2424	2430	2437	2438	2443	2461	2466	2467	2470	2474	2490
	2494	2495	2497	2517	2518	2520	2526	2533	2534	2538	2838	2850	2852	2855
	2874	2875	2877	2880	2896	2902	2903	2912	2915	2922	2942	2949	2950	2952
	2958	2962	2966	2972	2983	2985	2994	3017	3044	3057	3068	3081	3082	3083
	3089	3129	3179	3183	3184	3185	3186	3190	3206	3215	3236	3261	3310	3311
	3313	3319	3320	3324	3365	3369	3380	3384	3394	3407	3413	3422	3448	3780
	3804	3831	3832	3833	3836	3837	3838	3839	3852	3877	3918	3945	3965	3990
	4010	4036	4049	4074	4075	4076	4087	4111	4112	4113	4124	4151	4152	4153
	4163	4189	4190	4191	4202	4225	4242	4267	4284	4314	4351	4376	4377	4378
	4409	4437	4438	4439	4449	4471	4472	4473	4484	4507	4508	4509	4521	4547
	4548	4549	4565	4646	4656	4689	4730	4762	4774	4809	4811	4818	4826	4857
	4858	4869	4902	4966	4983	4997	5007	5017	5027	5036	5288	5289	5291	5293
	5295	5297	5299	5328	5330	5437	5449	5501	5502	5509	5522	5529	5542	5581
	5582	5589	5591											
AC5	49#	690	707	718	724	733	783	785	789	807	1065	1209	1273	1352
	1354	1358	1389	1499	1584	1668	1766	2392	2442	2897	2903	2904	2912	2913
	2916	2917	2921	2943	2946	2948	2949	3016	3069	3083	3086	3128	4682	4700
	4723	4741	4796	4812	4967	4982	4998	5008	5018	5028	5035	5059	5060	5068
	5069	5100	5102	5103	5111	5112	5157	5158	5166	5167	5173	5177	5215	5217
	5221	5225	5232	5241	5247	5249	5293	5294	5295	5297	5298	5299	5300	5302
	5304	5384	5385	5387	5389	5391	5392	5394	5396	5429	5430	5431	5474	5475
	5476	5511	5512	5519	5531	5532	5539	5589	5590	5591				
AC6	50#	2944	3002	3003	3006	3008	3015	3070	3127	4683	4684	4685	4699	4724
	4725	4726	4740	4968	4981	4999	5009	5019	5029	5034	5360	5361	5370	5378
	5380	5384	5391	5423	5425	5428	5431	5436	5448	5461				
AC7	51#	2945	2994	2995	3004	3014	3071	3078	3092	3126				
ACCARG	4038	4039	4041	4043	4057#									
ACCEPT	4033#	5462												
ACT00	552	571#												
ACT01	621	685#												
ACT01A	697	703#												
ACT01B	767	776#												
ACT01C	778	791#												
ACT01H	785#	790												
ACT01L	717	722#												
ACT01M	706	723	728#											
ACT01N	721	727	733#											
ACT01S	793	802	806#											
ACT01X	699	729	804#											
ACT02	823	871#												
ACT02A	890#	900												
ACT02B	893	899#												
ACT02C	902	910#												

ACT02E	897	906	916#	
ACT02S	918	922#		
ACT02X	912	921	923#	
ACT03	952	1060#		
ACT03A	1080#			
ACT03B	1103#	1108		
ACT03C	1107	1118#		
ACT03D	1120#	1130		
ACT03E	1123	1126	1130#	
ACT03F	1129	1136#		
ACT03G	1140	1145#		
ACT03I	1137	1150#		
ACT03J	1181	1195#		
ACT03S	1191	1201	1207#	
ACT03T	1204	1209#		
ACT03X	1113	1132	1146	1203#
ACT03Z	1144	1206#		
ACT04	1241	1398#		
ACT04S	1417#			
ACT04X	1416	1418#		
ACT06	1441	1494#		
ACT06A	1510	1517#		
ACT06B	1528#			
ACT06C	1530#	1540		
ACT06D	1533	1536	1540#	
ACT06E	1539	1546#		
ACT06F	1564	1567#		
ACT06H	1565	1572#		
ACT06J	1573	1576#		
ACT06S	1579	1583#		
ACT06X	1542	1568	1581#	
ACT07	1610	1663#		
ACT07A	1679	1686#		
ACT07B	1703#			
ACT07C	1705#	1715		
ACT07D	1708	1711	1715#	
ACT07E	1714	1721#		
ACT07F	1746	1749#		
ACT07G	1735	1744#		
ACT07H	1747	1754#		
ACT07J	1755	1758#		
ACT07S	1761	1765#		
ACT07X	1717	1750	1763#	
ACT08	1782	1803#		
ACT08A	1825	1835	1838#	
ACT08C	1816	1830#		
ACT08S	1826	1836	1841#	
ACT08X	1839#			
ACT09	1858	1890#		
ACT09A	1904	1912	1934#	
ACT09S	1933	1936#		
ACT10	1952	1985#		
ACT10A	2105#			

ACT10B	2012	2073	2087	2102	2107#				
ACT10S	2103	2110#							
ACT10X	2109#								
ACT11	2141	2181#							
ACT11B	2187	2196#							
ACT11S	2198#								
ACT11X	2193	2199#							
ACT12	2219	2225#							
ACT12X	2230#								
ACT3B1	1099	1112#							
ACTB	296#								
ACTC	289#								
ACTN	318#								
ADHBLA	2293#								
ADHBLK	2270#	5134	5266	5527	5547				
AICARG	3431	3433	3454#	3459					
AICLEN	3430	3454	3459#						
ANNOUN	91#	378							
BFSCA1	124#	659	716	718	892	894	1105	2405	5289
BFSCA2	126#	660	722	724	899	903	1106	2412	5291
BLINK	111#	2286	2318	2320	2325				
BOTEXE	235#	385	2281	[illegible]					
BUD0	334#	1170							
BUF1	5838#	5854							
BUF10	5838#	5854							
BUF100	5838#								
BUF11	5838#	5854							
BUF12	5838#	5854							
BUF13	5838#	5854							
BUF14	5838#	5854							
BUF15	5838#	5854							
BUF16	5838#	5854							
BUF17	5838#	5854							
BUF2	5838#	5854							
BUF20	5838#	5854							
BUF21	5838#	5854							
BUF22	5838#	5854							
BUF23	5838#	5854							
BUF24	5838#	5854							
BUF25	5838#	5854							
BUF26	5838#	5854							
BUF27	5838#	5854							
BUF3	5838#	5854							
BUF30	5838#	5854							
BUF31	5838#	5854							
BUF32	5838#	5854							
BUF33	5838#	5854							
BUF34	5838#	5854							
BUF35	5838#	5854							
BUF36	5838#	5854							
BUF37	5838#	5854							
BUF4	5838#	5854							
BUF40	5838#	5854							

BUF41 5838#
BUF42 5838#
BUF43 5838#
BUF44 5838#
BUF45 5838#
BUF46 5838#
BUF47 5838#
BUF5 5838# 5854
BUF50 5838#
BUF51 5838#
BUF52 5838#
BUF53 5838#
BUF54 5838#
BUF55 5838#
BUF56 5838#
BUF57 5838#
BUF6 5838# 5854
BUF60 5838#
BUF61 5838#
BUF62 5838#
BUF63 5838#
BUF64 5838#
BUF65 5838#
BUF66 5838#
BUF67 5838#
BUF7 5838# 5854
BUF70 5838#
BUF71 5838#
BUF72 5838#
BUF73 5838#
BUF74 5838#
BUF75 5838#
BUF76 5838#
BUF77 5838#
BUFAA 2465# 2468
BUFARG 420 4473 4474 4475 4477 4492#
BUFAV 694 2459#
BUFD 330# 1159
BUFF1 125# 720 895 2410
BUFF2 127# 726 904 2417
BUFFSZ 419 4468#
BUFL 315# 668 757 856 1017
BUFLEN 192# 670 704 759 771 782 797 1019 1030
BUFLIM 253# 662
BUFLST 2466 5849# 5856
BUFNAM 193# 695 711 719 725 764
BUFO 326# 1028 1165
BUFS 322# 739 881 1094
BUFSTE 2464 5856#
BUFT 284# 639 842
CONR 327#
C1NR 338#
CANTI1 3352# 3354

CANTIM	3009	3347#												
CAVAIL	5349	5608#												
CBKLEN	101#	2436	2532	5742										
CBLK1	5742#	5756												
CBLK10	5742#	5756												
CBLK11	5742#	5756												
CBLK12	5742#	5756												
CBLK13	5742#	5756												
CBLK14	5742#	5756												
CBLK15	5742#	5756												
CBLK16	5742#	5756												
CBLK17	5742#	5756												
CBLK2	5742#	5756												
CBLK20	5742#	5756												
CBLK21	5742#	5756												
CBLK22	5742#	5756												
CBLK23	5742#	5756												
CBLK24	5742#	5756												
CBLK25	5742#	5756												
CBLK26	5742#	5756												
CBLK27	5742#	5756												
CBLK3	5742#	5756												
CBLK30	5742#	5756												
CBLK31	5742#	5756												
CBLK32	5742#	5756												
CBLK33	5742#	5756												
CBLK34	5742#	5756												
CBLK35	5742#	5756												
CBLK36	5742#	5756												
CBLK37	5742#	5756												
CBLK4	5742#	5756												
CBLK40	5742#	5756												
CBLK5	5742#	5756												
CBLK6	5742#	5756												
CBLK7	5742#	5756												
CBLKAA	2364#	2367												
CBLKAV	2166	2358#	5422	5472										
CBLST	891	1104	1121	1531	1706	2365	2396	2494	2517	2902	2949	5059	5157	5288
	5360	5501	5581	5751#	5757									
CBLSTE	889	1102	1119	1529	1704	2363	2394	2492	2515	2900	2947	5057	5155	5286
	5358	5499	5579	5757#										
CBSTS	119#	1138	1142	2399	2497	2520	2903	2952	2983	2985	5173	5177	5221	5225
	5247	5249	5293	5295	5378	5380	5431	5476	5589	5591				
CHGOPC	550	618	820	967	1228	1435	1604	1780	1855	1949	2136	2210	2776#	
CHGOPX	2783	2787#												
CHKVCA	2901#	2905												
CHKVCB	2905#	2914	2918											
CHKVCS	2139	2893#												
CHKVCT	2907	2920#												
CHKVCX	2919	2921#												
CHNCNT	5619#	5628	5628#	5629	5629#	5630	5630#	5631	5631#	5632	5632#			
CHNTAB	3426	5627#												
CID	118#	1122	1175	1532	1707	2366	2397	2495	2518	2526	2950	2958	2966	2994

	5060	5158	5361	5428	5461	5473	5502	5582						
CIDLOC	209#	2763	2766											
CIPORT	217#	1076	1125	1505	1535	1548	1674	1710	1723	1808	1998			
CNOA	316#													
CNON	323#													
CN1A	331#													
CN1N	335#													
CNFARG	434	1905	2970	4549	4550	4551	4552	4553	4554	4555	4557	4573#		
CNFDA1	1857	1863#												
CNFDAT	533	1853#												
CNTLR	263	269#	4392	4395	4398									
CNTOLA	5410	5413#												
CNTOLB	5412	5422#												
CNTOLC	5414	5456#												
CNTOLD	5426	5457#												
CNTOLE	5443	5454	5461#											
CNTOLS	5339	5406#												
CNTOLX	5459	5464#												
CNTRLR	263#	3950												
CNTRR1	1951	1953#												
CNTRRD	534	1947#												
CONACC	5340	5469#												
CONARG	3879	3881	3882	3883	3885	3887	3896	3909	3910	3912	3915	3925#		
CONDAT	271#	3886	3929	4040	4059									
CONECT	2189	3875#												
CONFIG	432	1903	2186	2968	4544#									
CONFIX	4559	4562#												
CONIDC	207#	3962	5409											
CONIDR	208#	4006	5413											
CONREJ	5341	5486#												
CONT	132#	2130	3007	3282										
COUNT	96#	2245	2508	3001	3310	3350	3392	5085	5196	5737	5751	5775	5784	5785
	5786	5816	5834	5849										
CTPCOA	2126	2130#												
CTPCOB	2124	2136#												
CTPCOC	2129	2134	2138	2140	2147#									
CTPCON	535	2120#												
CTPCOX	2131	2142	2149#											
CTPPKT	114#	574	637	737	744	840	879	986	1081	1157	1275	1310	1331	1375
	1467	1555	1636	1730	1811	1873	1907	1967	2017	2158	2552	2584	2605	2627
	2648	2669	2743	2780	2847	3182	3209	3238	3786	3895	3900	3905	5086	5197
	5211													
CUREXE	237#	459	462	467	472	510	514	555	573	625	636	736	743	827
	839	878	947	957	985	1038	1080	1156	1235	1243	1274	1302	1309	1330
	1374	1443	1451	1466	1512	1554	1612	1620	1635	1681	1729	1786	1810	1861
	1872	1906	1957	1966	2016	2145	2157	2190	2215	2551	2563	2583	2604	2626
	2647	2668	2742	2779	3181	3208	3237	3318	3351	3785	4342			
CURSTS	58#	511	515	553	554	556	623	624	626	658	715	825	826	828
	939	971	972	973	1074	1133	1143	1174	1190	1200	1226	1238	1239	1252
	1253	1257	1258	1260	1287	1433	1449	1450	1452	1455	1503	1509	1572	1602
	1618	1619	1621	1624	1672	1678	1754	1784	1785	1787	1859	1860	1862	1955
	1956	1958	1996	2123	2125	2130	2143	2144	2146	2149	2213	2214	2216	2228
	2561	2562	2564	2916	3180	3207	3243	3282	3783	3795	3798	4323	4327	4341

```
         4343    5369    5374
CWAT1    3402    3503    3509#
CWAT11   3511    3516#
CWATSC   3501#   5137    5269    5318    5402    5549
CYCL60    198#   3550    3556    3566
DATAQ    4760#   5277
DATARG   4764    4765    4766    4768    4771    4782#
DATATE   4770    4773#
DATET1   3572    3576#
DATETM   3113    3571#
DECVER     20#     23      91    5706
DELA       285#  3240
DGM        135   1239    1253    1287    2876    3795    4323    5107    5118    5131    5237    5262
DGMCHN   3455    5629#   5629
DGMMSG    230#   5153    5214    5236    5261    5265
DGMSG     131#   1238    1239    1252    1253    5237    5262
DGMSRV    218#   5051
DIAAJ%   4921
DIABA%   4929
DIACI%   4930
DIAFC%   4922
DIAGAG   1547    1550    1558    1560    1562    1722    1725    1733    1742    1744    1809    1821    1822    1823
         1831    1832    1833    2003    2006    2007    2008    2010    2020    2025    2031    2037    2043    2049
         2055    2063    2066    2068    2069    2071    2078    2081    2082    2083    2085    2092    2095    2097
         2099    2100    4937#
DIAGF    4869#   4911
DIAGJ    1563    1745    1824    1834    2011    2072    2086    2101    4854#
DIAGS    4863    4868#
DIAGU    4860    4865#
DIAIA%   4916
DIAIF%   4925
DIANC%   4927
DIANP%   4915
DIANR%   4928
DIATO%   4931
DIAVC%   4926
DISARG   4113    4114    4115    4117    4132#
DISCON   2229    2528    4108#
DISIN1   3466    3472#
DISIN2   3470    3476#
DISINT    454    3464#
DLYADR   3311    3319    3352    3353    3394    3399    5784#
DLYGO    3245    3406#
DLYRBK    206#
DLYRTN   3320    3398    5786#
DLYTIM    205#   3241
DLYUPT   3324    3396    5785#
DMA       145#   1139    1141    5294
DMACHN   3457    5631#   5631
DMACMP    222#   5283
DMARD     221#   4735
DMAWRT    220#   4694
DMAXFO   5013#   5630
```

DMAXFA 5277# 5314
DMAXFB 5287# 5308
DMAXFC 5290 5293#
DMAXFD 5292 5308#
DMAXFR 4977 5020 5275#
DMAXFS 5306 5314#
DMAXFX 5282 5316#
DONE 138# 553 623 825 971 1258 1449 1618 1784 1859 1955 2143 2213 2561
4341 5512 5532
DOSCS 2804 3434 3640# 3800 3846 3913 3959 4003 4044 4081 4118 4158 4196 4234
4276 4344 4384 4443 4478 4515 4558 4616 4652 4692 4733 4769 4806
DOSCS1 3645 3663#
DOSCS2 3650 3654#
DSDG 341#
DUM1 4380 4393 4396 4399 4405 4415#
ENABL1 3523 3532#
ENABLE 389 3522#
ENAIN1 3483 3488#
ENAIN2 3486 3492#
ENAINT 491 3481#
ENDF 239# 363
ENTER 3056 3065#
ENTERC 3078# 3094
ENTERD 3081#
ENTERE 3085#
ENTERG 3088#
ENTERO 3079 3094#
ENTERP 3123# 3125
ENTERS 3099 3103#
ENTREN 3091#
ERRDI1 4866 4884#
ERRDI2 4862 4877#
ERRDI3 4894# 4896
ERRDI4 4880 4900#
ERRDI5 4893 4935#
ERRDIT 4894 4897 4915# 4935
ERRSC1 3676 3682#
ERRSC2 3688# 3690
ERRSC4 3680 3694#
ERRSCL 3687 3766#
ERRSCS 3435 3675# 3801 3847 3914 3960 4004 4045 4082 4119 4159 4197 4345 4385
4444 4479 4516 4693 4734
ERRSCT 3688 3691 3736# 3766
ERSTRT 3679 3719# 4879
EVENTE 4807 4911#
EVENTQ 4794# 5324
EVENTX 4820 4833#
EVTARG 4798 4799 4800 4801 4802 4803 4805 4808 4809 4835#
EVTTBL 5330 5337#
EXDLY 943 1231 1438 1607 3232#
EXDLY1 3247 3255#
EXDLY2 3242 3259#
EXDLY3 3257 3260#

EXDLY4 3251 3261#
EXECT 465 506#
EXTE 293# 1469 1557 1638 1732
EXWAI1 3284 3294#
EXWAI3 3296 3298#
EXWAI4 3290 3299#
EXWAIT 2192 3277#
FAKEBU 245#
FCMD 297#
FLINK 109# 468 2287 2288 2292 2319 2324 2326 2842 3003 3894 3899 3904
FSTS 319#
FUNCFL 254# 1145 1567 1749 1838 2127
FUNCST 524 548#
GA1CON 148# 5224
GA2CON 149# 5248
GAFLAG 223#
GANRES 147# 5176
GENC 300# 751 1317 1378
GENCON 190# 752 772 1318 1341
GENDGM 529 1220#
GENF 294# 650 746 1277 1312
GENFUN 189# 747 766 777 792 1313 1335 1346 1361
GENL 313# 2629
GENLEN 191# 1286 1323 1340 1351 1366 3210
GENMS1 1227 1234#
GENMS2 1230 1233 1242 1254#
GENMS4 1245 1257#
GENMS5 1232 1249 1260#
GENMS6 1237 1241#
GENMSG 528 1221#
GENRS1 1434 1441#
GENRS2 1437 1440 1442 1453#
GENRS3 1439 1448 1455#
GENRST 530 1428#
GENST1 1603 1610#
GENST2 1606 1609 1611 1622#
GENST3 1608 1617 1624#
GENSTR 531 1597#
GETACT 961 1150 1247 1409 1446 1576 1615 1758 2580#
GETARG 4610 4611 4612 4613 4615 4618 4628#
GETBUA 4617 4620#
GETBUF [illegible] 4608#
GETCID 2[illegible]3# 3884 3954 3998
GETGLN 1285 1322 2623#
GETINA 2948# 3013
GETINB 2951 2954 2990 3010 3013#
GETIND 3002# 3012
GETINE 3005 3011#
GETINF 460 2938#
GETINX 3014#
GETREP 1037 1301 1511 1680 2579#
GETTI2 3551 3560#
GOBIT 139# 464 554 624 826 972 1257 1450 1619 1785 1860 1956 2144 2214

	2562	3243	3406	5106	5117	5130	5235	5260	5303					
GPRS	1911	2796#												
GPRS1	2805	2808#												
GPRSB	2798	2799	2801	2803	2806	2814#								
GTINFO	144#	2953	2984	5430	5475									
HALTP	2254	2277	3314	3440	3443	3446	3474	3490	3534	3711#	5063	5090	5161	5201
	5312	5505	5585											
IFPIN1	3424	3438#												
IFPIN5	3436	3448#												
IFPINT	397	3419#												
INPDGO	4993#	5628												
INPDGA	5058#	5061												
INPDGB	5070	5083#												
INPDGC	5086#	5089												
INPDGD	5087	5099#												
INPDGE	5101	5111#												
INPDGF	5104	5129#												
INPDGH	5048#	5079	5109	5120	5135									
INPDGM	4975	5000	5046#											
INPDGX	5050	5137#												
INPMSO	5003#	5629												
INPMSA	5156#	5159												
INPMSB	5168	5194#												
INPMSC	5197#	5200												
INPMSD	5198	5210#												
INPMSE	5216	5241#												
INPMSF	5233	5259#												
INPMSG	4973	5010	5143#											
INPMSH	5145#	5181	5190	5226	5239	5250	5267							
INPMSJ	5175	5185#												
INPMSL	5223	5232#												
INPMSM	5234#	5246												
INPMSX	5147	5269#												
INTDMS	5001	5011	5021	5034#										
INTFLG	185#	478	480	3516										
INTTEN	4962#	5649												
KIL3RA	2493#	2500												
KIL3RB	2496	2500#												
KIL3RC	2509#	2512												
KIL3RD	476	2486#												
KIL3RE	2519	2522	2537#											
KIL3RF	2516#	2537												
KIL3RH	2533#	2535												
KIL3RX	2499	2538#												
LASTF	187#	946	950	968	1234	1240	1254	4332						
LBUFOF	228#	1167	1188	1197										
LDSTUS	565	648	675	804	854	861	911	920	1048	1070	1203	1386	1405	1483
	1524	1581	1652	1699	1763	1796	1839	1881	1899	1935	1975	1992	2109	212?
	2133	2171	2197	2212	2558	2665#	3250	3256	3289	3295				
LEVTAB	3426	3513	5636#											
LICARG	3947	3949	3951	3953	3955	3956	3958	3961	3973#					
LIRARG	3992	3994	3995	3997	3999	4000	4002	4005	4018#					
LISDI1	2218	2220#												

LISDIS 536 2207#
LISTNC 403 3942# 5411
LISTNR 404 3987# 5415
LIT.. 5860#
LK.LLC 3532
LK.LLS 3532
LK.LNE 3532
LK.LNP 3532
LOCALN 408 4434#
LOCARG 409 4439 4440 4442 4457#
LOCPRT 123# 1075 1124 1504 1534 1673 1709 1997 2915 2917 2972
LTLCRD 5343 5557#
M 5814# 5816 5822
MAINA 462# 473
MAINC 458 469 471 475#
MAINC1 479 490#
MAINLN 453# 481 497 3512
MAPBU1 620 627#
MAPBUF 525 616#
MATCH 163# 2227 2527
MAXDGM 225# 426 1289
MAXMSG 224# 423 1293
MAXOPC 102# 2748 2749 2782 3184 3789 3790 3837 3838
MDSND1 5500# 5503
MDSND2 5510 5513 5523 5529#
MDSND3 5530 5533 5543 5549#
MDSNDC 5342 5495#
MONTYP 197# 373 492 3039 3137 3328 3423 3465 3482 3522 3542 3571 3592 3594
3600 3610 3644 3675 3711 3713 4859
MOVE 295# 991 1083
MOVEB1 940 946#
MOVEB2 942 945 953 959 967#
MOVEB4 954 972#
MOVEB5 944 963 973#
MOVEBA 952#
MOVEBF 527 934#
MOVETY 196# 1084 1179
MSGCHN 3456 5630# 5630
MSGDGM 4316 4317 4318 4319 4320 4326 4359#
MSGSRV 219# 5148
MXDGM 99# 5814 5822
N 5736# 5741 5742 5742# 5750# 5755 5756 5756# 5774# 5779 5780 5780# 5815# 5822
5822# 5833# 5838 5838# 5848# 5853 5854 5854#
NMATCH 162# 5456
NMBYTS 3186 3791 3839 5664#
NOAC 305# 1813
NOBUFM 252# 910 1112
NOCONN 249# 1131 1541 1716
NODE 122# 1127 1537 1712 2912 2962
NODEUN 251# 1934 2196
NONSEN 257# 642 646 653 845 849 1032 1280 1472 1481 1641 1650 1879 1973
2105 2164
NORESO 256# 673 698 728 860 919 1022 1296 2107 2169 2194 3255 3294

NOTIMP 537 538 539 540 541 2548#
NOTMSG 3782 3784 3787 3792 3793 3797 3812#
NRESRC 164# 5424
NRSARG 3833 3834 3835 3840 3841 3845 3860#
NTIMPL 258# 996 2557
OFLINE 5344 5563#
OKSEND 5580# 5583
OKSEND 5346 5575#
OKTOSD 146# 5590
ONCHNS 3428 5618# 5628 5628# 5629 5629# 5630 5630# 5631 5631# 5632 5632#
ONETIM 137# 939 973 1226 1260 1433 1455 1509 1572 1602 1624 1678 1754 2123
2149
ONLINE 5345 5569#
OPCARY 576 578 594#
OPCHK 512 2740#
OPCO 278# 2553 2556 2744 2781 2786
OPCODE 194# 2554 2784 3183 3788 3836 4329
OTHCHN 3458 5632# 5632
OTHE 302# 1010 1088 1478 1518 1647 1687 1875 1969 2160
OTHERN 195# 1089 1128 1519 1538 1549 1688 1713 1724 1876 1902 1910 1970 2098
2161 2185 2913
OTHREO 5023# 5631
OTHREA 5324# 5331
OTHREV 4979 5030 5322#
OTHREX 5329 5333#
OURNOD 210# 411 798 1367
OVRDUE 134# 151# 2125 3412
P 60# 168 169 170 171 357 455 677 806 863 922 1050 1206 1207
1388 1417 1485 1583 1654 1765 1841 1883 1977 2173 2198 2807 2920 3189
3214 3259 3260 3298 3335 3641 3643 3651 3668 4350 4560 4619 4655 4868
4887 4907
POST 310# 1922
P10T11 735 2021 2026 2032 2038 2044 2050 2717#
P11T10 669 758 857 882 1018 1029 1095 1160 1166 1171 1741 2691#
P1ST 314# 1932
PCLEV1 5636 5642#
PCLEV2 5637 5643#
PCLEV3 5638 5644#
PCLREV 158# 271
PCLTYP 156# 271
PCLVER 157# 271
PIVECT 3438 3514 4971 5648# 5648 5653
PKTM 309#
PKTSZF 250#
PKTZ 306# 1002
PLIST 175# 175 357 455
PLISTE 175 177#
PLISTS 176#
PORT 301#
PRINT1 3601 3618#
PRINT2 3611 3619#
PRINT3 3620# 3624
PRINTA 3724 3728#

PRINTB	3725	3731#												
PRINTC	3584#	3707	4407	4828	4910									
PRINTD	3610#	4827												
PRINTO	3600#	3684	3698	3702	3706	4888	4904	4909						
PRINTT	379	549	617	819	935	1222	1429	1598	1779	1854	1948	2121	2208	2254
	2277	3314	3440	3443	3446	3474	3490	3534	3592#	3696	3700	3704	3729	3732
	3803	3850	3917	3964	4008	4047	4085	4122	4162	4200	4240	4282	4348	4388
	4404	4406	4447	4482	4519	4563	4623	4662	4697	4738	4779	4825	4878	4886
	4890	4898	4903	4906	4969	5047	5063	5090	5139	5144	5161	5201	5271	5276
	5312	5317	5323	5334	5356	5407	5470	5487	5496	5505	5558	5564	5570	5576
	5585	5597	5603	5609										
PRTBRK	5348	5602#												
PS.FAC	3441	3444												
PS.FOF	3472													
PS.FON	3488													
PS.TMS	3338													
QBDARG	4191	4192	4193	4195	4210#									
QBDGM	2851	4186#	5078											
QBMARG	4153	4154	4155	4157	4171#									
QBMSG	2853	4148#	5180	5189										
QUEBUF	2870#	5520	5540											
QUEBUX	2880#													
QUEUE	2835#	2879	5438	5439	5440	5441	5450	5451	5452	5453				
RB1	5780	5822#												
RB10	5780	5822#												
RB100	5780	5822#												
RB101	5780	5822#												
RB102	5780	5822#												
RB103	5780	5822#												
RB104	5780	5822#												
RB105	5780	5822#												
RB106	5780	5822#												
RB107	5780	5822#												
RB11	5780	5822#												
RB110	5780	5822#												
RB111	5780	5822#												
RB112	5780	5822#												
RB113	5780	5822#												
RB114	5780	5822#												
RB115	5780	5822#												
RB116	5780	5822#												
RB117	5780	5822#												
RB12	5780	5822#												
RB120	5780	5822#												
RB121	5780	5822#												
RB122	5780	5822#												
RB123	5780	5822#												
RB124	5780	5822#												
RB125	5780	5822#												
RB126	5780	5822#												
RB127	5780	5822#												
RB13	5780	5822#												
RB130	5780	5822#												

RB131	5780	5822#
RB132	5780	5822#
RB133	5780	5822#
RB134	5780	5822#
RB135	5780	5822#
RB136	5780	5822#
RB137	5780	5822#
RB14	5780	5822#
RB140	5780	5822#
RB141	5780	5822#
RB142	5780	5822#
RB143	5780	5822#
RB144	5780	5822#
RB145	5780	5822#
RB146	5780	5822#
RB147	5780	5822#
RB15	5780	5822#
RB150	5780	5822#
RB151	5780	5822#
RB152	5780	5822#
RB153	5780	5822#
RB154	5780	5822#
RB155	5780	5822#
RB156	5780	5822#
RB157	5780	5822#
RB16	5780	5822#
RB160	5780	5822#
RB161	5780	5822#
RB162	5780	5822#
RB163	5780	5822#
RB164	5780	5822#
RB165	5780	5822#
RB166	5780	5822#
RB167	5780	5822#
RB17	5780	5822#
RB170	5780	5822#
RB171	5780	5822#
RB172	5780	5822#
RB173	5780	5822#
RB174	5780	5822#
RB175	5780	5822#
RB176	5780	5822#
RB177	5780	5822#
RB2	5780	5822#
RB20	5780	5822#
RB200	5780	5822#
RB201	5780	5822#
RB202	5780	5822#
RB203	5780	5822#
RB204	5780	5822#
RB205	5780	5822#
RB206	5780	5822#
RB207	5780	5822#

RB21	5780	5822#
RB210	5780	5822#
RB211	5780	5822#
RB212	5780	5822#
RB213	5780	5822#
RB214	5780	5822#
RB215	5780	5822#
RB216	5780	5822#
RB217	5780	5822#
RB22	5780	5822#
RB220	5780	5822#
RB221	5780	5822#
RB222	5780	5822#
RB223	5780	5822#
RB224	5780	5822#
RB225	5780	5822#
RB226	5780	5822#
RB227	5780	5822#
RB23	5780	5822#
RB230	5780	5822#
RB231	5780	5822#
RB232	5780	5822#
RB233	5780	5822#
RB234	5780	5822#
RB235	5780	5822#
RB236	5780	5822#
RB237	5780	5822#
RB24	5780	5822#
RB240	5780	5822#
RB241	5780	5822#
RB242	5780	5822#
RB243	5780	5822#
RB244	5780	5822#
RB245	5780	5822#
RB246	5780	5822#
RB247	5780	5822#
RB25	5780	5822#
RB250	5780	5822#
RB251	5780	5822#
RB252	5780	5822#
RB253	5780	5822#
RB254	5780	5822#
RB255	5780	5822#
RB256	5780	5822#
RB257	5780	5822#
RB26	5780	5822#
RB260	5780	5822#
RB261	5780	5822#
RB262	5780	5822#
RB263	5780	5822#
RB264	5780	5822#
RB265	5780	5822#
RB266	5780	5822#

RB267	5780	5822#
RB27	5780	5822#
RB270	5780	5822#
RB271	5780	5822#
RB272	5780	5822#
RB273	5780	5822#
RB274	5780	5822#
RB275	5780	5822#
RB276	5780	5822#
RB277	5780	5822#
RB3	5780	5822#
RB30	5780	5822#
RB300	5780	5822#
RB301	5780	5822#
RB302	5780	5822#
RB303	5780	5822#
RB304	5780	5822#
RB305	5780	5822#
RB306	5780	5822#
RB307	5780	5822#
RB31	5780	5822#
RB310	5780	5822#
RB311	5780	5822#
RB312	5780	5822#
RB313	5780	5822#
RB314	5780	5822#
RB315	5780	5822#
RB316	5780	5822#
RB317	5780	5822#
RB32	5780	5822#
RB320	5780	5822#
RB321	5780	5822#
RB322	5780	5822#
RB323	5780	5822#
RB324	5780	5822#
RB325	5780	5822#
RB326	5780	5822#
RB327	5780	5822#
RB33	5780	5822#
RB330	5780	5822#
RB331	5780	5822#
RB332	5780	5822#
RB333	5780	5822#
RB334	5780	5822#
RB335	5780	5822#
RB336	5780	5822#
RB337	5780	5822#
RB34	5780	5822#
RB340	5780	5822#
RB341	5780	5822#
RB342	5780	5822#
RB343	5780	5822#
RB344	5780	5822#

RB345	5780	5822#
RB346	5780	5822#
RB347	5780	5822#
RB35	5780	5822#
RB350	5780	5822#
RB351	5780	5822#
RB352	5780	5822#
RB353	5780	5822#
RB354	5780	5822#
RB355	5780	5822#
RB356	5780	5822#
RB357	5780	5822#
RB36	5780	5822#
RB360	5780	5822#
RB361	5780	5822#
RB362	5780	5822#
RB363	5780	5822#
RB364	5780	5822#
RB365	5780	5822#
RB366	5780	5822#
RB367	5780	5822#
RB37	5780	5822#
RB370	5780	5822#
RB371	5780	5822#
RB372	5780	5822#
RB373	5780	5822#
RB374	5780	5822#
RB375	5780	5822#
RB376	5780	5822#
RB377	5780	5822#
RB4	5780	5822#
RB40	5780	5822#
RB400	5780	5822#
RB401	5822#	
RB402	5822#	
RB41	5780	5822#
RB42	5780	5822#
RB43	5780	5822#
RB44	5730	5822#
RB45	5780	5822#
RB46	5780	5822#
RB47	5780	5822#
RB5	5780	5822#
RB50	5780	5822#
RB51	5780	5822#
RB52	5780	5822#
RB53	5780	5822#
RB54	5780	5822#
RB55	5780	5822#
RB56	5780	5822#
RB57	5780	5822#
RB6	5780	5822#
RB60	5780	5822#

RB61 5780 5822#
RB62 5780 5822#
RB63 5780 5822#
RB64 5780 5822#
RB65 5780 5822#
RB66 5780 5822#
RB67 5780 5822#
RB7 5780 5822#
RB70 5780 5822#
RB71 5780 5822#
RB72 5780 5822#
RB73 5780 5822#
RB74 5780 5822#
RB75 5780 5822#
RB76 5780 5822#
RB77 5780 5822#
RBUFOF 229# 1172 1187 1198
RCDARG 4269 4270 4271 4272 4273 4275 4279 4292# 5049 5052 5074 5075 5083
RCMARG 4227 4228 4229 4230 4231 4233 4237 4250# 5146 5149 5178 5179 5185 5186
5194
RCVDGE 4277 4279#
RCVDGM 4264# 5048
RCVMSE 4235 4237#
RCVMSG 4222# 5145
REALBU 244#
REDARG 4726 4727 4728 4729 4730 4732 4746#
REDATA 1189 4723#
REFE 281#
REJARG 4076 4077 4078 4080 4095#
REJECT 4071# 5458
REPC 290# 2586
REQ1 120# 2331 2332 2424 5068 5100 5102 5166 5215 5217 5297 5384 5509 5542
REQ2 121# 2333 2334 2430 5069 5103 5111 5112 5167 5232 5241 5299 5391 5522
5529
RESOFS 103# 2555 2747 3185
RESPFG 188# 5416 5442 5464
RH1 2246 2258 2509 2510 3002 5086 5099 5197 5210 5780#
RH10 5780#
RH100 5780#
RH101 5780#
RH102 5780#
RH103 5780#
RH104 5780#
RH105 5780#
RH106 5780#
RH107 5780#
RH11 5780#
RH110 5780#
RH111 5780#
RH112 5780#
RH113 5780#
RH114 5780#
RH115 5780#

RH116	5780#
RH117	5780#
RH12	5780#
RH120	5780#
RH121	5780#
RH122	5780#
RH123	5780#
RH124	5780#
RH125	5780#
RH126	5780#
RH127	5780#
RH13	5780#
RH130	5780#
RH131	5780#
RH132	5780#
RH133	5780#
RH134	5780#
RH135	5780#
RH136	5780#
RH137	5780#
RH14	5780#
RH140	5780#
RH141	5780#
RH142	5780#
RH143	5780#
RH144	5780#
RH145	5780#
RH146	5780#
RH147	5780#
RH15	5780#
RH150	5780#
RH151	5780#
RH152	5780#
RH153	5780#
RH154	5780#
RH155	5780#
RH156	5780#
RH157	5780#
RH16	5780#
RH160	5780#
RH161	5780#
RH162	5780#
RH163	5780#
RH164	5780#
RH165	5780#
RH166	5780#
RH167	5780#
RH17	5780#
RH170	5780#
RH171	5780#
RH172	5780#
RH173	5780#
RH174	5780#

RH175	5780#
RH176	5780#
RH177	5780#
RH2	5780#
RH20	5780#
RH200	5780#
RH201	5780#
PH202	5780#
RH203	5780#
RH204	5780#
RH205	5780#
RH206	5780#
RH207	5780#
RH21	5780#
RH210	5780#
RH211	5780#
RH212	5780#
RH213	5780#
RH214	5780#
RH215	5780#
RH216	5780#
RH217	5780#
RH22	5780#
RH220	5780#
RH221	5780#
RH222	5780#
RH223	5780#
RH224	5780#
RH225	5780#
RH226	5780#
RH227	5780#
RH23	5780#
RH230	5780#
RH231	5780#
RH232	5780#
RH233	5780#
RH234	5780#
RH235	5780#
RH236	5780#
RH237	5780#
RH24	5780#
RH240	5780#
RH241	5780#
RH242	5780#
RH243	5780#
RH244	5780#
RH245	5780#
RH246	5780#
RH247	5780#
RH25	5780#
RH250	5780#
RH251	5780#
RH252	5780#

RH253	5780#
RH254	5780#
RH255	5780#
RH256	5780#
RH257	5780#
RH26	5780#
RH260	5780#
RH261	5780#
RH262	5780#
RH263	5780#
RH264	5780#
RH265	5780#
RH266	5780#
RH267	5780#
RH27	5780#
RH270	5780#
RH271	5780#
RH272	5780#
RH273	5780#
RH274	5780#
RH275	5780#
RH276	5780#
RH277	5780#
RH3	5780#
RH30	5780#
RH300	5780#
RH301	5780#
RH302	5780#
RH303	5780#
RH304	5780#
RH305	5780#
RH306	5780#
RH307	5780#
RH31	5780#
RH310	5780#
RH311	5780#
RH312	5780#
RH313	5780#
RH314	5780#
RH315	5780#
RH316	5780#
RH317	5780#
RH32	5780#
RH320	5780#
RH321	5780#
RH322	5780#
RH323	5780#
RH324	5780#
RH325	5780#
RH326	5780#
RH327	5780#
RH33	5780#
RH330	5780#

RH331	5780#
RH332	5780#
RH333	5780#
RH334	5780#
RH335	5780#
RH336	5780#
RH337	5780#
RH34	5780#
RH340	5780#
RH341	5780#
RH342	5780#
RH343	5780#
RH344	5780#
RH345	5780#
RH346	5780#
RH347	5780#
RH35	5780#
RH350	5780#
RH351	5780#
RH352	5780#
RH353	5780#
RH354	5780#
RH355	5780#
RH356	5780#
RH357	5780#
RH36	5780#
RH360	5780#
RH361	5780#
RH362	5780#
RH363	5780#
RH364	5780#
RH365	5780#
RH366	5780#
RH367	5780#
RH37	5780#
RH370	5780#
RH371	5780#
RH372	5780#
RH373	5780#
RH374	5780#
RH375	5780#
RH376	5780#
RH377	5780#
RH4	5780#
RH40	5780#
RH400	5780#
RH41	5780#
RH42	5780#
RH43	5780#
RH44	5780#
RH45	5780#
RH46	5780#
RH47	5780#

```
RH5       5780#
RH50      5780#
RH51      5780#
RH52      5780#
RH53      5780#
RH54      5780#
RH55      5780#
RH56      5780#
RH57      5780#
RH6       5780#
RH60      5780#
RH61      5780#
RH62      5780#
RH63      5780#
RH64      5780#
RH65      5780#
RH66      5780#
RH67      5780#
RH7       5780#
RH70      5780#
RH71      5780#
RH72      5780#
RH73      5780#
RH74      5780#
RH75      5780#
RH76      5780#
RH77      5780#
RHBLEN     100#  2248  2338  2511  3011  5088  5199  5780
RHBLKA    2246#  2249
RHBLKB    2247   2258#
RHBLKV    2242#  2841  3893  3898  3903
RHBPER     97#   2245  2508  3001  5085  5196  5775  5816
RMCBLA    2395#  2441
RMCBLB    2437#  2439
RMCBLC    2398   2401  2441#
RMCBLG    2406   2412#
RMCBLH    2413   2424#
RMCBLK     466    475  2385#
RMCBLM    2430#
RMDISC    5347   5596#
RMHBLK    2306#  2426  2432  5514  5534
RMRBLA    2339#  2341
RMRBLB    2314   2343#
RMVARG    4648   4649  4651  4668#
RMVBUA    4653   4655#
RMVBUF     917   2407  2414  4643#
R^CNT      112#   948   958  1039  1236  1244  1303  1444  1513  1613  1682
RQREFU     255#
RSPDR      264    270#
RSPNDR     264#  3880  3948  3993
S..CAI    3443#
S..CCO    5585#
S..CFC    5505#
```

```
S..CFD     5090#
S..CFM     5161#
S..CII     3440#
S..CLJ     3534#
S..CST     3446#
S..DBF     3314#
S..EON     3490#
S..NCD     5312#
S..NCI     5063#
S..NRH     2254#
S..NRM     5201#
S..POF     3474#
S..RNQ     2277#
SC%DGA     4974
SC%DTA     4976
SC%EVA     4978
SC%MSA     4972
SCALBF      226#   884    916   1097   1098   1186   1195
SCARBF      227#  1162   1185   1196
SCOFFS     3441   3502   4971   5648#
SCSBLK      203#  3701
SCSCAL      200#  3642   3705
SCSERC      201#  3656   3669   3683   3686   3720
SCSFCN      202#  3640   3697
SEDARG     4685   4686   4687   4688   4689   4691   4705#
SEDATA     1199   4682#
SEN04      1255   1415   3203#
SEN04C     3215#
SEND        557    627    829    969   1453   1622   1788   1863   1953   2147   2217   3176#
SENDA      3190#
SENDMA     4333   4344#
SENDMC     4328   4351#
SENDMD     3187   3212   4311#
SENIMA     3799   3802#
SENIMP     2559   3777#
SENNRA     3849#
SENNRS     3828#  5077   5188
SETTI1     3329   3338#
SETTI2     3335#  3341
SETTI6     3312   3318#
SETTI7     3311#  3313
SETTIM     3246   3283   3310#
SIXCID      110#  2191
SPEAR       477   3039#
SPEARA     3050   3054#
SPEARE     3077   3122   5696#
SPEARF      184#  3049   3055
SPEART     2979   3078   3123   3124   5692#   5696
SPEARX     3052   3057#
SPOFST      105#  2976   3082
SPRBLN      348#  3102
SPRCIN      350#   437   3086
SPRNOD      352#  3089
```

SQ%CVD 4629
SSACR% 3739
SSACS% 3740
SSARG% 3738
SSCBF% 3754
SSCPN% 3741
SSCRB% 3758
SSCUB% 3759
SSDCF% 3751
SSDQE% 3756
SSEQE% 3757
SSICI% 3747
SSIDM% 3762
SSIFC% 3737
SSINN% 3744
SSNBQ% 3749
SSNFB% 3752
SSNFC% 3745
SSNNK% 3743
SSNPC% 3742
SSNPV% 3736
SSNSB% 3760
SSPSF% 3755
SSQBF% 3753
SSRCF% 3750
SSRQE% 3748
SSSCP% 3763
SSSTL% 3764
SSTMS% 3761
SSUNS1 1788#
SSUNSO 532 1778#
SSVNO% 3746
STAARG 4509 4510 4511 4512 4514 4529#
STAR 317# 1693
START 357# 3715 5863
STAT 286# 2671
STATSI 4504#
STATUS 113# 463 511 515 556 626 828 1452 1621 1787 1862 1958 2146 2216
2330 2564 2843 3006 3008 3407 3413 4343 5105 5108 5116 5119 5129 5132
5234 5238 5259 5263 5302 5304 5387 5389 5394 5396 5511 5531
STPARG 2960 4378 4379 4381 4383 4419#
STPOL1 4394 4397 4400 4403#
STPOL2 4401 4409#
STPOLL 2959 4373#
STRACT 965 1045 1152 1251 1383 1411 1575 1578 1757 1760 2602#
STRGLN 2645#
STRREP 2601#
STRTAD 213# 1694 1739
STRTF 183# 361 362
SUCCES 243# 564 1069 1404 1523 1698 1795 1898 1991 2132 2211 3249 3288
SYSAC2 216# 3104 3142
SYSE02 3108 3114 5702#
SYSE10 415 436 5704#

SYSE12	3085	3088	3092	3101	3109	5709#	5715						
SYSERE	5713#												
SYSERT	3141	5700#	5713	5715									
SYSHLF	3103	5715#											
SYSJYS	3118	3137#											
THIRD	143#	2498	2521	2904	2989	5174	5222	52[illegible]5	5475				
TIMCHN	3332	5628#	5628										
TIMELN	104#	3048											
TIMESP	215#	443	3047	3054									
TIMIN1	3394#	3400											
TIMIN7	3395	3397	3400#										
TIMINT	3367	3382	3392#										
TIMR10	3362#	5654											
TIMR20	3377#	5627											
TIOFFS	3401	3444	5653#										
TOPEXE	234#	383	384	456	457	470							
TRACE	179#	548	616	818	934	1221	1428	1597	1778	1853	1947	2120	2207	3802
	3849	3916	3963	4007	4046	4084	4121	4161	4199	4239	4281	4347	4387	4446
	4481	4518	4562	4622	4661	4696	4737	4778	4819	4969	5046	5138	5143	5270
	5275	5316	5322	5333	5355	5406	5469	5486	5495	5557	5563	5569	5575	5596
	5602	5608												
TSTTBL	513	524#												
UNMAP1	822	829#												
UNMAPB	526	818#												
UNSACT	214#	1814												
UTIM1	3543	3549#												
UTIME	441	3045	3321	3393	3542#									
VAL00	551	563#												
VAL01	619	632#												
VAL01A	645	648#												
VAL01C	652	658#												
VAL01D	661	667#												
VAL01G	641	644#												
VAL01S	672	677#												
VAL01X	643	647	654	663	675#									
VAL02	821	835#												
VAL02A	848	854#												
VAL02G	844	847#												
VAL02S	859	863#												
VAL02X	846	850	861#											
VAL03	941	981#												
VAL03A	995	1001#												
VAL03B	1004	1009#												
VAL03C	1016#													
VAL03F	1021	1027#												
VAL03G	993	1005	1012	1032#										
VAL03H	1031	1037#												
VAL03S	1046	1050#												
VAL03X	997	1023	1033	1048#										
VAL04	1229	1268#												
VAL04A	1279	1285#												
VAL04B	1288	1293#												
VAL04C	1291	1296#												

```
VAL04D    1290    1294    1301#
VAL04E    1336    1345#
VAL04F    1354#   1359
VAL04G    1347    1360#
VAL04H    1362    1374#
VAL04S    1384    1388#
VAL04X    1281    1297    1386#
VAL06     1436    1463#
VAL06A    1471    1477#
VAL06S    1480    1485#
VAL06X    1473    1483#
VAL07     1605    1632#
VAL07A    1640    1646#
VAL07S    1649    1654#
VAL07X    1642    1652#
VAL08     1781    1794#
VAL09     1856    1869#
VAL09S    1878    1883#
VAL09X    1881#
VAL10     1950    1963#
VAL10S    1972    1977#
VAL10X    1975#
VAL11     2137    2154#
VAL11A    2163    2166#
VAL11S    2168    2173#
VAL11X    2165    2171#
VCBRKA    5359#   5362
VCBRKB    5386    5391#
VCBRKC    5364    5393    5402#
VCBRKN    5338    5355#   5565    5598    5604
VCGONE     136#    152#   2400    3798    4327    5374    5379    5388    5395
VEDIT       19#      23     91
WAITF      186#     453    490    3510    3515
WAITIM     211#
WAITRB     212#
WAITT      133#    1136    1143    1190    1200
WATOVR    3281    3412#
%CNDTM    3576
%CNMNT     367
%CNSTS    3560
%CNSUP    3552
.DGCTR    2002    2062    2077    2091
.DGENB    1820    1830
.DGRST    1546
.DGSTR    1721
.FHSLF    3331    3425    3468    3484    3524    3654    3721
.LBACC    4037    4057
.LBCON    3878    3925
.LBCSP    4377    4419
.LBDIS    4112    4132
.LBEVT    4797    4803    4835
.LBGDE    4763    4782
.LBGLN    4438    4457
```

```
.LBLIS   3946  3973  3991  4018
.LBQRD   4190  4210
.LBQRM   4152  4171
.LBRBS   4472  4492
.LBRCD   4548  4555  4573
.LBRDG   4268  4292
.LBREJ   4075  4095
.LBREQ   4725  4746
.LBRMG   4226  4250
.LBRPS   2800  2814
.LBSMG   3781  3812  3832  3860  4315  4359
.LBSND   4684  4705
.LBSTS   4508  4529
.LBUMP   4647  4668
.PCSCS   3441  5651
.PCTMR   3444  5656
.PRIOU   3602  3612  3719
.PSVIS   4971
.PSVOP   3514
.SIDGA   3455
.SIDMA   3457
.SIMSA   3456
.SIPAN   3458
.SQADC   3909
.SQAFB   4155  4193
.SQAMC   3896
.SQAPT   3787  3835  4318
.SQARB   4229  4237  4271  4279  5049  5075  5083  5146  5179  5186  5194
.SQBDN   4381
.SQBID   4766  4771
.SQBNA   4611  4612  4613  4618
.SQCDA   4041
.SQCDT   3887
.SQCID   3784  3834  4039  4077  4114  4154  4192  4228  4270  4317  4379  4510  4550  4686
         4727  4765  4799  4808  5052  5074  5149  5178  5185
.SQDFL   4230  4272
.SQDIS   4115
.SQDPN   3882  3951  3995
.SQESB   4800  4801
.SQEVT   4802  4809
.SQFLG   3793  3841  4320
.SQFST   4511
.SQLCI   3956  3961  4000  4005
.SQLDG    424  4475
.SQLEN   2801  3431  3782  3833  3879  3947  3992  4038  4076  4113  4153  4191  4227  4269
         4316  4378  4439  4473  4509  4549  4610  4648  4685  4726  4764  4798
.SQLMG    421  4474
.SQLNN    410  4440
.SQLPN    435  2971
.SQLPT   3792  3840  4319
.SQLRP   4231  4273
.SQNAM   4649
.SQOFS   4689  4730
```

```
.SQOSB    4551
.SQRCI    3910
.SQREJ    4078
.SQRNM    4688    4729
.SQRPN    2798
.SQRPS    2799    2806
.SQSAD    4554
.SQSBI    2961
.SQSBR    4512
.SQSNM    4687    4728
.SQSPN    3881    3949    3994
.SQSYS    3883    3885    3915    3953    3955    3997    3999
.SQVCS    4552    4553
.SSACC    4042
.SSAIC    3432
.SSCON    3911
.SSCSP    4382
.SSDIS    4116
.SSEVT    4804
.SSGDE    4767
.SSGLN    4441
.SSLIS    3957    4001
.SSMAP    4614
.SSQRD    4194
.SSQRM    4156
.SSRBS    4476
.SSRCD    4556
.SSRDG    4274
.SSREJ    4079
.SSREQ    4731
.SSRMG    4232
.SSRPS    2802
.SSSDG    3796    3844    4324
.SSSMG    3794    3842    4322
.SSSND    4690
.SSSTS    4513
.SSUMP    4650
.TIMEL    3331
```

AIC% 3429
ASSINT 5620# 5627 5628 5629 5630 5631
BUF 5829# 5838
BUFLS 5842# 5853 5854
CBLK 5731# 5741 5742
CBLS 5744# 5755 5756
CRASH 78# 2254 2277 3314 3440 3443 3446 3474 3490 3534 5063 5090 5161 5201
5312 5505 5585
DEBRK% 3388 5041
DEBRK. 3373 4988
DIAG% 4861
DIAG. 4865
DIR% 3469
EIR% 3485
EPCAP% 3528
ERCAL 3724 3725
ERJMP 3144 3334 3526 3529 3605 3615 3650 4862
ERSTR% 3723
EXIT 3714
GET 171# 516 517 549 566 583 584 617 678 679 680 681 807 808
809 810 811 812 819 864 865 866 867 923 924 925 926 935
974 975 976 1051 1052 1053 1054 1209 1210 1211 1212 1213 1214 1222
1261 1262 1263 1389 1390 1391 1392 1393 1394 1418 1419 1420 1429 1456
1457 1458 1486 1487 1488 1584 1585 1586 1587 1588 1589 1598 1625 1626
1627 1655 1656 1657 1766 1767 1768 1769 1770 1771 1779 1797 1842 1843
1844 1845 1846 1854 1884 1885 1886 1936 1937 1938 1939 1940 1948 1978
1979 1980 2110 2111 2112 2121 2150 2174 2175 2176 2199 2200 2201 2208
2220 2230 2231 2259 2260 2293 2294 2295 2343 2344 2345 2346 2373 2374
2375 2442 2443 2444 2445 2446 2447 2474 2475 2476 2538 2539 2540 2541
2542 2565 2566 2567 2591 2592 2611 2612 2634 2635 2654 2655 2672 2673
2674 2703 2704 2729 2730 2750 2751 2787 2788 2789 2808 2809 2855 2856
2857 2858 2880 2881 2882 2883 2884 2921 2922 2923 2924 2925 3014 3015
3016 3017 3018 3019 3020 3021 3057 3058 3059 3060 3126 3127 3128 3129
3130 3131 3132 3145 3146 3190 3191 3192 3193 3215 3216 3217 3218 3261
3262 3263 3264 3265 3299 3300 3301 3356 3357 3369 3370 3371 3372 3384
3385 3386 3387 3448 3449 3450 3451 3476 3492 3504 3517 3546 3557 3625
3803 3804 3805 3806 3807 3850 3852 3853 3854 3855 3917 3918 3919 3920
3964 3965 3966 3967 3968 4008 4010 4011 4012 4013 4047 4049 4050 4051
4052 4085 4087 4088 4089 4090 4122 4124 4125 4126 4127 4162 4163 4164
4165 4166 4200 4202 4203 4204 4205 4240 4242 4243 4244 4245 4282 4284
4285 4286 4287 4348 4351 4352 4353 4354 4388 4409 4410 4411 4412 4447
4449 4450 4451 4452 4482 4484 4485 4486 4487 4519 4521 4522 4523 4524
4563 4565 4566 4567 4568 4620 4623 4656 4657 4658 4659 4662 4697 4699
4700 4738 4740 4741 4774 4775 4776 4779 4812 4813 4814 4829 4830 4831
4869 4870 4871 4872 4891 4969 4981 4982 4983 4984 4985 4986 4987 5034
5035 5036 5037 5038 5039 5040 5047 5139 5144 5271 5276 5317 5323 5334
5356 5407 5470 5487 5496 5558 5564 5570 5576 5597 5603 5609
GETER% 3655
GETTAB 368 3553 3561 3577
GO 168# 379 389 397 403 404 408 419 432 441 454 460 465 466
475 476 477 491 512 513 549 550 551 552 557 565 617 618
619 621 627 648 669 675 694 705 735 758 804 819 820 821
823 829 854 857 861 882 911 917 920 935 941 943 952 961

```
                965   967   969  1018  1029  1037  1045  1048  1070  1095  1150  1152  1160  1166
               1171  1189  1199  1203  1222  1228  1229  1231  1241  1247  1251  1255  1285  1301
               1322  1383  1386  1405  1409  1411  1415  1429  1435  1436  1438  1441  1446  1453
               1483  1511  1524  1563  1575  1576  1578  1581  1598  1604  1605  1607  1610  1615
               1622  1652  1680  1699  1741  1745  1757  1758  1760  1763  1779  1780  1781  1782
               1788  1796  1824  1834  1839  1854  1855  1856  1858  1863  1881  1899  1903  1911
               1935  1948  1949  1950  1952  1953  1975  1992  2011  2021  2026  2032  2038  2044
               2050  2072  2086  2101  2109  2121  2128  2133  2136  2137  2139  2141  2147  2166
               2171  2186  2189  2192  2197  2208  2210  2212  2217  2219  2229  2254  2277  2407
               2414  2426  2432  2528  2558  2559  2804  2841  2851  2853  2879  2959  2968  3009
               3045  3056  3113  3118  3187  3212  3246  3250  3256  3283  3289  3295  3314  3321
               3367  3382  3393  3398  3434  3435  3440  3443  3446  3474  3490  3503  3534  3551
               3624  3679  3684  3696  3698  3700  3702  3704  3706  3707  3800  3801  3803  3846
               3847  3850  3884  3893  3898  3903  3913  3914  3917  3954  3959  3960  3964  3998
               4003  4004  4008  4044  4045  4047  4081  4082  4085  4118  4119  4122  4158  4159
               4162  4196  4197  4200  4234  4240  4276  4282  4344  4345  4348  4384  4385  4388
               4404  4406  4407  4443  4444  4447  4478  4479  4482  4515  4516  4519  4558  4563
               4616  4623  4652  4662  4692  4693  4697  4733  4734  4738  4769  4779  4806  4825
               4827  4828  4878  4879  4886  4888  4890  4898  4903  4904  4906  4909  4910  4969
               4980  5000  5010  5020  5030  5047  5048  5063  5077  5078  5090  5134  5137  5139
               5144  5145  5161  5180  5188  5189  5201  5266  5269  5271  5276  5277  5312  5317
               5318  5323  5324  5330  5334  5356  5407  5411  5415  5422  5438  5439  5440  5441
               5450  5451  5452  5453  5458  5462  5470  5472  5487  5496  5505  5514  5520  5527
               5534  5540  5547  5558  5564  5570  5576  5585  5597  5603  5609
GTAD%          3573
HALT           5301
HALTF%         3712
HIBER           495
LOCK           3533
NOP             955
NOUT%          3604  3614
OUTCHR         3626
OUTSTR         3595  3682  3685  3692
PIINI.         3439
PISYS.         3442  3445  3473  3489
PITMR.         3339
PRVERS          87#    91
PSOUT%         3593  3678
PUT             170#  508   509   549   563   571   572   617   632   633   634   635   685   686
                687   688   689   690   819   835   836   837   838   871   872   873   874   935
                936   937   938   981   982   983   984  1060  1061  1062  1063  1064  1065  1222
               1223  1224  1225  1268  1269  1270  1271  1272  1273  1398  1399  1400  1429  1430
               1431  1432  1463  1464  14    1494  1495  1496  1497  1498  1499  1598  1599  1600
               1601  1632  1633  1634  16    1664  1665  1666  1667  1668  1779  1794  1803  1804
               1805  1806  1807  1854  1     1870  1871  1890  1891  1892  1893  1894  1948  1963
               196-  1965  1985  1986  1987  2121  2122  2154  2155  2156  2181  2182  2183  2208
               2209  2225  2226  2242  2243  2270  2271  2272  2306  2307  2308  2309  2358  2359
               2360  2387  2388  2389  2390  2391  2392  2459  2460  2461  2486  2487  2488  2489
               2490  2548  2549  2550  2580  2581  2602  2603  2623  2624  2645  2646  2665  2666
               2667  2691  2692  2717  2718  2740  2741  2776  2777  2778  2796  2797  2835  2836
               2837  2838  2870  2871  2872  2873  2874  2893  2894  2895  2896  2897  2938  2939
               2940  2941  2942  2943  2944  2945  3041  3042  3043  3044  3065  3066  3067  3068
               3069  3070  3071  3139  3140  3176  3177  3178  3179  3203  3204  3205  3206  3232
```

```
                3233    3234    3235    3236    3277    3278    3279    3347    3348    3362    3363    3364    3365    3377
                3378    3379    3380    3419    3420    3421    3422    3464    3481    3501    3509    3544    3549    3622
                3777    3778    3779    3780    3803    3828    3829    3830    3831    3850    3875    3876    3877    3917
                3942    3943    3944    3945    3964    3987    3988    3989    3990    4008    4033    4034    4035    4036
                4047    4071    4072    4073    4074    4085    4108    4109    4110    4111    4122    4148    4149    4150
                4151    4162    4186    4187    4188    4189    4200    4222    4223    4224    4225    4240    4264    4265
                4266    4267    4282    4311    4312    4313    4314    4348    4373    4374    4375    4376    4388    4434
                4435    4436    4437    4447    4468    4469    4470    4471    4482    4504    4505    4506    4507    4519
                4544    4545    4546    4547    4563    4608    4623    4643    4644    4645    4646    4662    4682    4683
                4697    4723    4724    4738    4760    4761    4762    4779    4794    4795    4796    4821    4822    4823
                4854    4855    4856    4857    4884    4962    4963    4964    4965    4966    4967    4968    4969    4993
                4994    4995    4996    4997    4998    4999    5003    5004    5005    5006    5007    5008    5009    5013
                5014    5015    5016    5017    5018    5019    5023    5024    5025    5026    5027    5028    5029    5047
                5139    5144    5271    5276    5317    5323    5334    5356    5407    5470    5487    5496    5558    5564
                5570    5576    5597    5603    5609
RBLK            5811#   5822
RESET%           371
RHBLK           5768#   5779    5780
RPCAP%          3525
RTN              169#    519     549     559     567     585     617     629     682     813     819     831     868     927
                 935     977    1055    1215    1222    1264    1395    1421    1429    1459    1489    1590    1598    1628
                1658    1772    1779    1790    1798    1847    1854    1865    1887    1941    1948    1959    1981    2113
                2121    2151    2177    2202    2208    2221    2232    2261    2296    2347    2376    2449    2477    2543
                2568    2593    2613    2636    2656    2675    2705    2731    2752    2767    2790    2810    2859    2885
                2926    3022    3040    3061    3133    3138    3147    3194    3219    3266    3302    3336    3340    3358
                3408    3414    3452    3477    3493    3505    3518    3530    3535    3547    3558    3567    3574    3579
                3596    3606    3616    3627    3652    3657    3670    3726    3803    3808    3850    3856    3917    3921
                3964    3969    4008    4014    4047    4053    4085    4091    4122    4128    4162    4167    4200    4206
                4240    4246    4282    4288    4348    4355    4388    4413    4447    4453    4482    4488    4519    4525
                4563    4569    4623    4624    4662    4664    4697    4701    4738    4742    4779    4780    4833    4873
                4969    4970    5047    5139    5140    5144    5271    5272    5276    5317    5319    5323    5334    5335
                5356    5407    5465    5470    5477    5487    5488    5496    5558    5559    5564    5570    5571    5576
                5592    5597    5603    5609    5610
SCS%            3649
SCS.            3666
SIR%            3427
SYERR%          3143
TIME%           3545
TIMER%          3333
TRMSG             66#    548     613     818     934    1221    1428    1597    1778    1853    1947    2120    2207    3802
                3849    3916    3963    4007    4046    4084    4121    4161    4199    4239    4281    4347    4387    4446
                4481    4518    4562    4622    4661    4696    4737    4778    4969    5046    5138    5143    5270    5275
                5316    5322    5333    5355    5406    5469    5486    5495    5557    5563    5569    5575    5596    5602
                5608
WAIT%            493
```